www.freesoftwaremagazine.com Issue 3 - April, 2005

Free Software MAGAZINE

FOCUS:

FREE AND PROPRIETARY SOFTWARE TOGETHER

(CHRIS J. KARR)

PROMOTING FREE SOFTWARE ON NON FREE PLATFORMS

(EDWARD MACNAGHTEN)

THE RISK OF MIXING FREE AND NON-FREE

(DANIEL O. ESCASA)

WHEN FREE MEETS PROPRIETARY

(SAQIB ALI)

XML: WYSIWYG to WYSIWYM

A BRIEF LOOK AT XML DOCUMENT AUTHORING

(DAVID M. BERRY)

Free as in "free speech" or free as in "free labour"?

A PHILOSOPHICAL ENQUIRY INTO FREE CULTURE

(TOM JACKIEWICZ)

The importance of LDAP

THE PAST, PRESENT AND FUTURE OF A BATTERED PROTOCOL

(TOM CHANCE)

Guerrilla marketing part two

GETTING GOOD PRESS COVERAGE

(MARTIN C. BROWN)

Running BSD on PowerPC/PPC

USING AN ALTERNATIVE UNIX ON AN ALTERNATIVE PLATFORM

COVER BY WWW.QREATE.NET

Are your systems *OPEN* for business?

Our customers' are!

For over 10 years, OCI has been helping organizations build their mission critical infrastructure with open architectures using standards-based OO technologies.

OCI provides the following commercial-grade products to the open-source community:

- **ACE** - The high performance C++ portability layer combining socket-level performance with type-safe programming, across a wide range of operating systems.
- **TAO** - An open-source C++ CORBA 2.6 implementation with the widest range of services and span of platforms of any ORB.
- **JacORB** - The open-source Java CORBA 2.4 implementation.
- **MPC** - An open-source cross-platform build tool - "script once, build many"
- **OVATION** - A tool for instrumenting & visualizing CORBA systems behavior.

Can't wait to go open? OCI offers a comprehensive set of migration support services that will help your organization minimize risk, cost, and time when moving to these open-source solutions.

To learn more about these products & services, visit: www.theaceorb.com

Product Development • Consulting • Educational Services

Object Computing, Inc. (OCI)

St. Louis, MO Headquarters: +1 (314) 579-0066
Phoenix, AZ Office: +1 (480) 752-0042

www.ociweb.com

OBJECT COMPUTING, INC.

New vision of Java

Want to buy the **Software 2.0** magazine, please visit our shop at **www.shop.software.pl**

Free Software MAGAZINE

Contents

Issue 3, April 2005

EDITORIAL

Thank you! 7

Thanks to all of the subscribers who are supporting our project

STARTERS

Book review: Knoppix Hacks *by Kyle Rankin* 9

by Jeremy Turner

FOCUS

Promoting free software on non-free platforms 10

by Chris J. Karr

Why developing free software for proprietary platforms benefits the free software community

The risk of mixing free and non-free 15

by Edward Macnaghten

In a world where anything (even SCO…) is possible, we have to remember that risks are taken when free and non-free software are mixed

When free meets proprietary 20

by Daniel O. Escasa

Three free applications on proprietary systems, three proprietary applications on free systems

TECH WORLD

XML: WYSIWYG to WYSIWYM 25

by Saqib Ali

A brief look at XML document authoring

A laptop, a coffee, and disaster recovery 29

by John Locke

Why you should have an effective backup strategy

The importance of LDAP 33

by Tom Jackiewicz

The past, present and future of a battered protocol

Running BSD on PowerPC/PPC 37

by Martin C Brown

Using an alternative Unix on an alternative platform

WORD WORLD

Guerrilla marketing part two 42

by Tom Chance

Getting good press coverage

Open source is a market force 47

by Malcolm D. Spence

How open source has changed and continues to change the IT market, and the behaviours of those who target it

Free software gets small 55

by Christian Einfeldt

The mouse that might roar

Free as in “free speech” or free as in “free labour”? 59

by David M. Berry

A philosophical enquiry into free culture

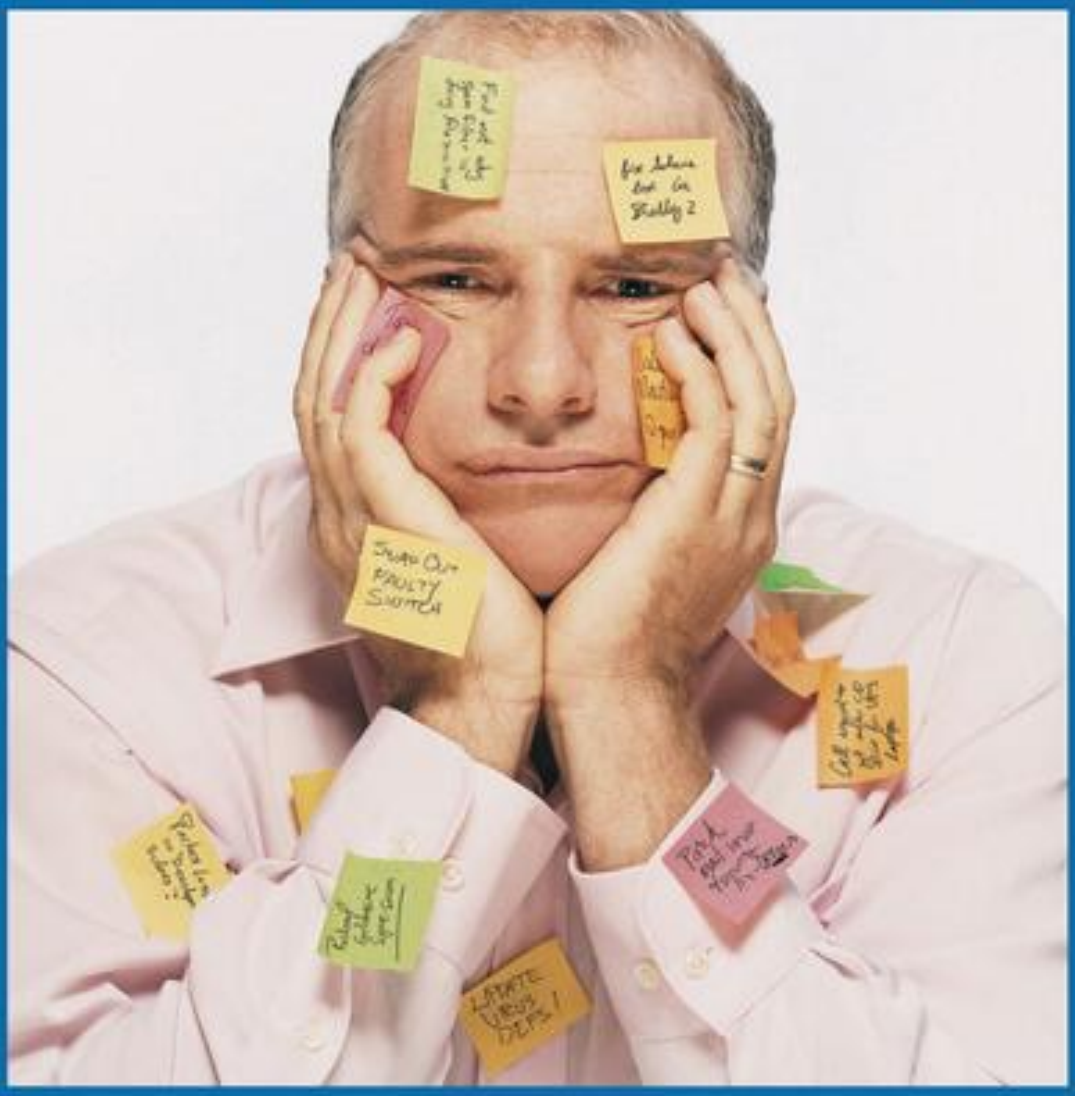
Richard Stallman’s blog 63

by Richard Stallman

Selected entries from Richard’s blog (http://www.fsf.org/blogs/rms/).

A PLANT NEEDS WATER TO GROW
Free Software
MAGAZINE
BY SUBSCRIBING YOU WILL BE SUPPORTING A MAGAZINE WHICH BELIEVES IN FREE SOFTWARE. ALL OUR ARTICLES ARE RELEASED UNDER THE GNU FREE DOCUMENTATION LICENSE, ENHANCING EXISTING INFORMATION ON FREE SOFTWARE.
SUBSCRIBE NOW!
WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE
FREE SOFTWARE MAGAZINE
NEEDS SUBSCRIBERS TO PROSPER!

Losing track of things?



There is a better way.

RT.

Enterprise-grade issue tracking.
Open source customizability.
Commercial support.

Download it now!

Best Practical Solutions offers training, support and custom development services. Check out our website or contact us for details.

www.bestpractical.com
sales@bestpractical.com

BEST PRACTICAL ™

Thank you!

Thanks to all of the subscribers who are supporting our project

Thank you!

This "thank you" is dedicated to all of the subscribers who are now reading issue 3 of Free Software Magazine. You have decided that it was worthwhile paying money for Free Software Magazine and have placed your trust in our project.

I appreciate your help, and I promise that we will do our very best to not disappoint.

As you probably already know, each month FSM will cover a specific subject with 2 or 3 "focus" articles. This month's focus is on mixing free software and proprietary systems. I think that Chris, Eddy and Daniel have done a terrific job dealing with this sometimes, controversial topic. I hope that their articles will help to clarify the issues surrounding this topic and help you to understand your choices. I'm sure that you'll enjoy reading their articles as much as I did.

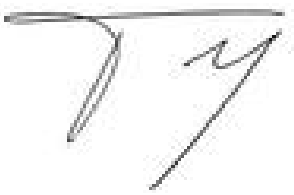
At the beginning of this project, more than four months ago, I was told by several people (some of them "experts") that it would be very hard to create a "proper magazine", rather than a rough collection of (hopefully well written) blogs. Some of these responses could have been classified as constructive criticism, while others didn't leave much space to hope. Well, I must say, they were right: creating a magazine is indeed a very hard task. Content is only one small part of the task. There's administration, subscriptions, composition, graphic design, and printing to name a few; I confess that when an issue is posted to the subscribers, it feels a bit like we've performed a miracle.

However, reading the articles published here in issue 3, and seeing the quality produced by our first-class authors, I can proudly say that, while it was hard, we have done it! Free Software Magazine is a collection of valuable articles, our publishing company is very well organised - even though it's spread across the globe - and things are happening; the miracle is occurring every month, thanks to your help and your trust.

If you have any ideas, comments or complaints about this magazine, please don't hesitate to contact me. Free Software Magazine is *your* magazine, and I will answer your email as soon as I receive it.

Again, thank you for making this project possible.

© 2005 by Tony Mobily
Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.



Free Software Magazine is a magazine by **The Open Company Partners Inc**, 90 Main St. Road Town, Tortola BVI

EDITOR IN CHIEF
Tony Mobily (t.mobily@)

TECHNICAL EDITORS
Clare James (c.james@)
Pancrazio De Mauro (p.demauro@)
Gianluca Insolvibile (g.insolvibile@)

EDITORS
Anna Dymitr Hawkes (a.dymitrhawkes@)
Dave Guard (d.guard@)

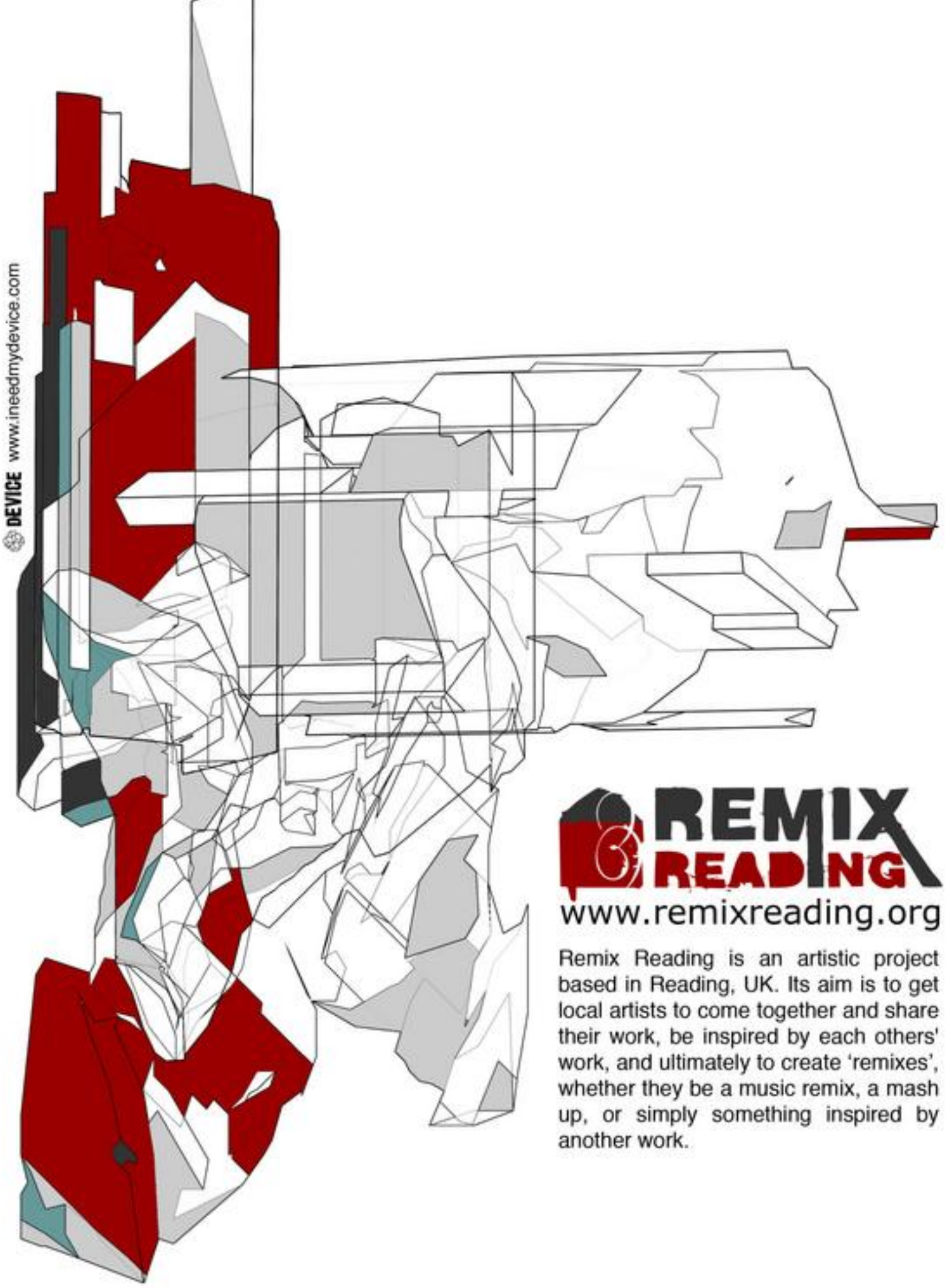
TECHS
Gianluca Pignalberi (LaTeX class and magazine generation) (g.pignalberi@)
Gian Maria Ricci (RTF to XML converter using VBA) (gm.ricci@)

GRAPHIC DESIGN
Alan Sprecacenere (Web, cover and advertising design) (a.sprecacenere@)
Tony Mobily, Gianluca Pignalberi, Alan Sprecacenere (Magazine design)

THIS PROJECT EXISTS THANKS TO
Donald E. Knuth, Leslie Lamport, People at TeX Users Group TUG (http://www.tug.org)

Every listed person is contactable by email. Please just add freesoftwaremagazine.com to the person's username in parentheses.

For copyright information about the contents of Free Software Magazine, please see the section "Copyright information" at the end of each article. Unless the license is applied immediately with the writing "(The following license is effective immediately)", for six weeks after publication you may not reproduce or retransmit the article, in whole or in part, in any manner, without the prior written consent of the author; six weeks after publication, the license at the end of each article applies.



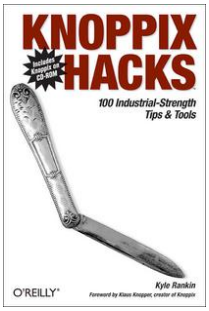
REMIX READING

www.remixreading.org

Remix Reading is an artistic project based in Reading, UK. Its aim is to get local artists to come together and share their work, be inspired by each others' work, and ultimately to create 'remixes', whether they be a music remix, a mash up, or simply something inspired by another work.

Book review: Knoppix Hacks *by Kyle Rankin*

Jeremy Turner



Knoppix is a live-CD Linux distribution which comes with X Window and some of the most exciting and useful programs in the free software world ready for use.

Like the famous Swiss Army Knife, "Knoppix Hacks" is an invaluable device. It has the best tips, tricks, and tools, along with information on other Knoppix-like systems. It contains common pitfalls and ways around them, most of which I had to discover by trial and error. Knoppix has quirks like mounting hard drive partitions read-only by default, but Mr. Rankin not only provides warning, but also provides guidance on how to mount read-write.

The contents

"Knoppix Hacks" covers several areas, including: booting, using, and tweaking Knoppix, installing Linux, using Knoppix in real-world situations, repairing both Linux and Windows installations, Knoppix variants, and re-mastering your own Knoppix. Of the 100 listed "hacks", I consider 42 of them to be essential reading. These include using Knoppix to setup temporary Samba or Apache servers when the main server is unavailable, and other hacks like repairing (or repartitioning) Linux. The book also teaches how to edit the Windows registry from Knoppix. The remaining hacks deal with using Knoppix: how to use cheat codes, connect to the internet, install applications, and create a persistent home directory. This information is necessary for understanding the later hacks: how to go wardriving, migrate to software RAID, and repair damaged file systems. I really enjoyed reading the contributed "hacks" about Knoppix-like systems Morphix, Gnoppix, and DamnSmallLinux.

This link (`http://www.oreilly.com/catalog/knoppixhks/`) is the book's home page at O'Reilly. It contains PDFs of 5 hacks, along with the complete table of contents.

Who's this book for?

I would recommend "Knoppix Hacks" to any Linux user. It will help newer Knoppix users avoid pitfalls. However, the more experienced Linux users will find "Knoppix Hacks" to be a valuable addition to their tech toolbox, along with Knoppix itself. And when you come across someone who's never heard of Linux, this just might be the book and CD to introduce them with.

Pros

There were no major errors that I could find with the text. At one point, I did find that the author gave instructions for the user to unmount a mounted file system, and first change out of the directory. The "cd" command didn't change the prompt as expected to reflect that the user would now be in the /home/knoppix directory. If the user was still in the directory of a mounted file system, the file system could not be unmounted. Having to nitpick this closely shows how well this book was written. Mr. Rankin includes a few humorous comments along the way so that the reader isn't overloaded or bored to tears.

Cons

None

Like the famous Swiss Army Knife,
"Knoppix Hacks" is an invaluable device.
It has the best tips, tricks, and tools,
along with information on other
Knoppix-like systems

In short

Title	Knoppix Hacks
Author	Kyle Rankin
Publisher	O'Reilly
ISBN	0596007876
Year	2004
Pages	336
CD included	Yes
Mark (out of 10)	9

Copyright information

© 2005 by Jeremy Turner

This article is made available under the "Attribution-NonCommercial-NoDerivs" Creative Commons License 2.0 available from http://creativecommons.org/licenses/by-nc-nd/2.0/.

Promoting free software on non-free platforms

Why developing free software for proprietary platforms benefits the free software community

Chris J. Karr

In a recent discussion on the Slashdot web site (http://slashdot.org/), free software users and advocates raised the question of whether the KDE project should be ported to the Microsoft Windows platform. Advocates for porting the KDE desktop environment made the argument that porting KDE to Windows would enable a new population of users to experience the software and that this exposure would entice these new users to seek out and adopt free software for use in their daily computing lives. Opponents of porting KDE believe that since Microsoft controls the underlying platform, software like KDE will never be able to compete on a fair playing field and that the time and effort spent in porting the software would be wasted when the vendor eventually decided to exclude free software competitors. While some free software advocates are right to be concerned, current realities support the position that free software on non-free platforms should be promoted.

Motives for developing free software can be divided into ideological and pragmatic groups

Before becoming heavily involved in the dispute, one must pay attention to the motivations of those producing and using free software. Some developers approach free software development from a point of view of promoting fundamental rights. Richard Stallman, of the GNU Project, is the most well known member of this group, with his philosophies on software development and use. This group believes that there are fundamental rights that grant software users the freedom to modify the software they use to better serve their needs and that users should possess the freedoms to share and distribute these changes. Unrestricted access to source code is central to this philosophy and closed-source software with restrictions on copying and distribution is an anathema to those holding these views. Members of this group produce free software as a means to establishing alternatives to the non-free platforms and applications.

In contrast to the ideologically-driven developers, another group produces free software for pragmatic reasons. These pragmatic developers are less interested in software politics and are more interested in using the free software development model and free software licenses to achieve a practical goal of their own - such as selling services, using free software as a competitive tool in the marketplace, or simply making the software available to the widest user base possible. Many adopt the free software development model in order to build a community and promote collaboration and they solicit contributions from the community to drive further development and maintain users' interest in the software.

While there are other motives for developing free software, most can be divided into ideological and pragmatic groups. The people who primarily view free software as a philosophical movement are ideological participants, while those who release free software as a means to another end are

pragmatic participants. Despite these fundamental differences, both camps benefit when free software is available on both free and non-free platforms.

"Don't let the perfect be the enemy of the good."

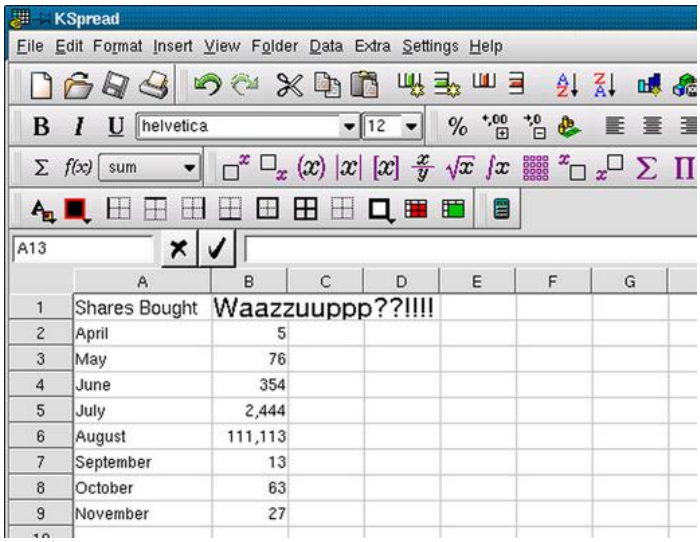
While pragmatic developers do not object to developing free software for non-free platforms, ideological developers often do. Ideological participants have a strong interest in the creation of an environment with complete freedom, so that users can utilize free software in all facets of their computing lives. To accomplish this goal, ideological participants believe that it is advantageous if the free platform grew and innovated independently of non-free platforms. In their view, it makes sense for the free software platform to evolve into a distinct environment that has its own advantages, not found in non-free platforms. Porting free software to non-free platforms minimizes the distinction between free and non-free platforms and the presence of free software functionality on non-free platforms provides less of an incentive for users to switch.

By any reasonable measure, the ideological developers have already achieved a remarkable goal. Using a free Unix operating system with software such as OpenOffice.org, the Mozilla internet browsers, and free software media players, an ideological user can go through their everyday tasks without using any non-free software.

However, when interacting with non-ideological users, a pure free software platform is not always sufficient. Some web sites use non-free technologies such as Flash. Document exchange is troublesome as non-free office formats complicate interoperability with free software. New Digital Rights Management (DRM) technologies exclude free media software.

While ideological developers have succeeded in creating a pure free software platform, they now face a much more insurmountable problem – motivating non-ideological users to exclusively adopt free software. While free software use has grown greatly in recent years, the market of non-free software and platforms still dwarfs that of free software. Ideological developers prefer a computing landscape where all layers of the computing platform are free, but they face an uphill battle in convincing non-ideological users to abandon their non-free software. While ideological users were

Fig. 1: Will the KOffice suite be a competitor to OpenOffice.org and Microsoft Office?



patient and contributed time and effort to help free software reach its current level of functionality, the non-ideological user values current availability and ease-of-use over philosophical purity. Furthermore, there are no free software applications approaching the level of functionality and polish found in some popular non-free software (such as Quicken and Photoshop), and non-ideological users will not abandon their non-free platforms while suitable free software replacements are absent.

While ideological developers have succeeded in creating a pure free software platform, they now face a much more insurmountable problem – motivating non-ideological users to exclusively adopt free software

Thus ideological users are unable to convert the multitudes of non-free software users, but they benefit every time their non-ideological counterparts use free software. Each Windows or Mac OS X user that installs FireFox and browses the web using a free browser is one less user that web sites can target with non-free technologies such as ActiveX.

A bank website can afford to ignore the one percent of their clients who are pure free software users, but they cannot afford to ignore twenty percent of their clients who

Fig. 2: Firefox is winning the hearts and minds of Mac and Windows users everywhere

are on non-free software platforms using free software web browsers. Ideological users benefit similarly when more users adopt free office suites instead of Microsoft Office. Proponents of pure free software stacks benefit from network effects each time a user of a non-free platform adopts a free replacement for a non-free application.

In addition to their own personal use, ideological users further benefit from free software adoption when promoting and educating users about freedom and software. Trying to make a case that a Windows XP user should switch to a free browser is difficult if there is no way to easily demonstrate such a browser on a non-free platform. It is difficult to sell the idea that KDE is a wonderful desktop environment when the user is required to install Linux to try it out. In this situation, the availability of free software on non-free platforms serves as an advertisement that free software is of high quality and that it can replace non-free applications.

Furthermore, as users adopt more free alternatives in place of non-free applications, the task of converting the user from a hybrid free and non-free platform to a pure free software platform becomes much simpler. Since the user is already familiar with applications, the switching costs and education required for the new free platform is greatly reduced. A user on Windows who primarily uses free software is easier to convert over to Linux, as it's simply a matter of teaching the user how to use Linux instead of having to teach them to use an entirely new suite of software and Linux. While applications exclusive to non-free platforms will prevent some users from being able to switch from a mixed system to a pure one, the adoption of high-quality free applications like Firefox and OpenOffice.org is shrinking that pool of users. As more free software replacements are made available to non-free users, the task of the free software advocate becomes much easier when convincing users to switch.

Finally, the idea that people willing and interested in writing free software should be discouraged from writing code that runs on non-free platforms is hypocritical and troubling when it comes from ideological users. One of the core tenets of the free software movement is the idea that users are free to expand and share their applications. Whenever ideological users discourage developers from free development of any kind, it violates the core principle that developers possess the freedom to develop as they feel is appropriate. Simply put, an ideological user has no right to complain when another developer targets a non-free platform with a particular application, when the ideological user is perfectly capable of donating their time or other resources to achieve the same end on a free platform. Ideological users are not owed any consideration when such a generous gift is given freely.

Making the most of non-free platforms

For the pragmatic free software developer, developing for non-free platforms provides many advantages. In terms of software engineering, a diversity of supported platforms supports encourages developers to implement their applications in more modular and portable ways. When working on non-free platforms, the pragmatic developer also has the opportunity to interoperate and collaborate with best-of-breed commercial and free software applications. Developers can craft their applications to take advantage of features unique to non-free platforms, such as using media codecs and security features. Non-free platforms also provide a stable set of libraries and base applications that free software developers can depend upon – including XML parsers, secure internet libraries, and HTML rendering engines. Finally, developers targeting non-free platforms can take advantage of these platforms' users' willingness to purchase software in order to fund the creation and further development of the free software.

As a direct result of competitive markets, non-free platforms

Fig. 3: The free software VLC media player is a popular alternative to Quicktime and Windows Media players

typically have larger libraries of high-quality end-user applications than their free counterparts. This is not to disparage free software, but when software is purchased, there is a reasonable expectation that the purchase includes not only the basic functionality, but also things like documentation, support, and polish. Furthermore, the market is much more favorable to applications that focus on usability and consistency, whereas free software typically focuses on the quantity of features. When developing for non-free platforms, free software developers can interact with non-free applications in a variety of ways. In Mac OS X, this can be accomplished via Services and AppleScript, while Windows employs a COM-based architecture. For example, an aspiring developer can create a media player that uses native functionality (such as QuickTime or Windows Media) that allows the playback of more formats than similar applications found on free software platforms. This is becoming increasingly important as content owners adopt DRM technology.

In addition to interoperating with non-free applications on non-free platforms, the platforms themselves offer unique features that developers may wish to use. For a developer wishing to create a free software Mac OS X application, the availability of the Cocoa framework provides the developer a set of tools and programming interfaces that are more fully featured and easier to work with than comparable free software GUI toolkits. While similar frameworks such as GNUStep struggle to keep up, Cocoa developers can take advantage of KHTML-based web rendering and new search technologies in their applications with a few lines of Objective-C or Java code.

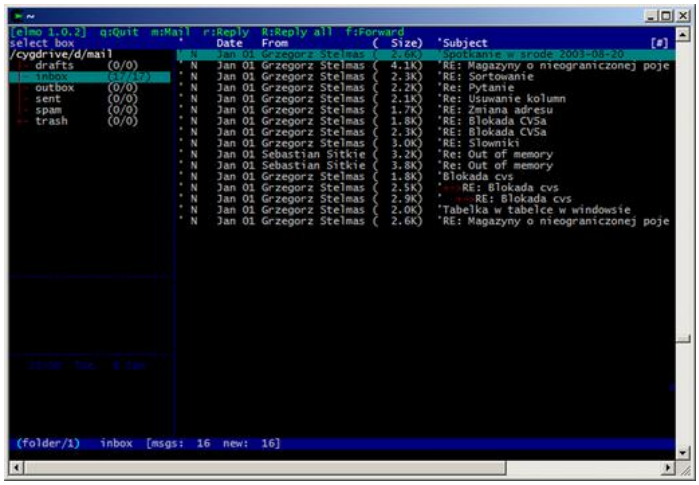
Non-free platforms offer unique features that free software developers may wish to use

Developing applications on non-free platforms also provides developers a limited amount of freedom from "dependency hell". Free platforms are often designed to allow users maximum flexibility to customize the system for uses ranging from slimmed-down routers and servers to full-featured desktop systems for daily use. In contrast, non-free platforms often include more functionality in the base installs and while users complain about bloat, this "bloat" allows developers to provide more advanced applications that are easier to install and maintain. Developers can distribute packages and installation instructions that are simpler than the typical free software application with the standard list of packages and library dependencies. This is mainly true with basic functionality, although developers may find it necessary to specify or bundle dependent packages when dealing with new or advanced features not present on the non-free system. For example, while a free software package may require that a particular XML parsing library is present on free platforms, the same package may also be able to make use of the non-free systems' own XML routines.

In terms of software engineering, developing applications that function on both free and non-free software can improve the underlying structure of application by requiring a certain amount of abstraction from the underlying platform. When the platform-specific code is separated from the platform-nonspecific code, the program can be more easily ported to new platforms by simply implementing the platform-specific code. Furthermore, separating the code into modular components can improve debugging and optimizations as the platform-specific code can be adapted to particular platforms' strengths and quirks. This tendency toward modularity assists both pragmatic and ideological developers, as the software can be adapted to new platforms as they emerge.

Finally, from the point of view of funding the creation of

Fig. 4: The Cygwin environment provides a Unix-like environment on Microsoft Windows.


[c]mo 1.0.2] q:Quit m:Mail r:Reply R:Reply all f:Forward
select box
/cygdrive/d/mail
drafts (0/0)
inbox (0/16)
outbox (0/0)
sent (0/0)
spam (0/0)
trash (0/0)
Date From (Size) *Subject [#]
Jan 01 Grzegorz Stelmas (4.1K) *RE: Magazyny o nieograniczonej poje
Jan 01 Grzegorz Stelmas (2.3K) *RE: Sortowanie
Jan 01 Grzegorz Stelmas (2.2K) *Re: Pytanie
Jan 01 Grzegorz Stelmas (2.1K) *Re: Usuwanie kolumn
Jan 01 Grzegorz Stelmas (1.7K) *RE: Zmiana adresu
Jan 01 Grzegorz Stelmas (1.8K) *RE: Blokada CVSa
Jan 01 Grzegorz Stelmas (2.3K) *RE: Blokada CVSa
Jan 01 Grzegorz Stelmas (3.0K) *RE: Slowniki
Jan 01 Sebastian Sitkie (3.2K) *Re: Out of memory
Jan 01 Sebastian Sitkie (3.8K) *Re: Out of memory
Jan 01 Grzegorz Stelmas (1.8K) *Blokada cvs
Jan 01 Grzegorz Stelmas (2.5K) *RE: Blokada cvs
Jan 01 Grzegorz Stelmas (2.9K) *RE: Blokada cvs
Jan 01 Grzegorz Stelmas (2.0K) *Tabelka w tabelce w windowsie
Jan 01 Grzegorz Stelmas (2.6K) *RE: Magazyny o nieograniczonej poje
(folder/1) inbox [msgs: 16 new: 16]

free software, there are cultural advantages to developing free software that includes non-free platforms. Since users on non-free platforms are accustomed to purchasing software, free software developers can take advantage of this social trait by selling easily installed versions of the software and materials such as documentation and manuals. Some free software developers have supplemented their income by making boxed versions of their software available to new users while allowing more advanced users to download the source online and compile the software locally.

Conclusion

Ideological and pragmatic developers approach the question of whether free software should be ported to non-free platforms differently. Ideological developers are generally against the idea for philosophical reasons. Pragmatic developers are generally agnostic with respect to the question and are more open to developing for non-free platforms. Ideological users need to realize that the vision of a pure free software environment for all users will never become a reality while non-ideological users exist, because non-ideological users will not spend the time or energy to write or wait for free alternatives to all of their non-free applications.

However, despite the continued existence of hybrid free and non-free systems, ideological users have much to gain by supporting the porting of free software on non-free platforms by limiting the amount of influence of non-free applications and file formats. Furthermore, by providing free software to users on non-free platforms, ideological users can better educate new users about the philosophy of free software and the freedom to distribute and modify their software. Pragmatic developers also gain when developing for non-free platforms, because developing for different types of platforms allows the pragmatic developers to make the most of features and functionality provided by commercial developers. Furthermore, by embracing a diverse set of platforms, pragmatic developers can leverage that experience to create more modular and portable software.

The idea that people devoted to free software should discourage the development and use of free software on non-free platforms makes little sense, considering what is to gain by developing free software for non-free platforms.

Copyright information

© 2005 by Chris J. Karr

This article is made available under the “Attribution-NonCommercial-NoDerivs” Creative Commons License 2.0 available from http://creativecommons.org/licenses/by-nc-nd/2.0/.

About the author

Chris Karr is a software developer for Northwestern University. He maintains a personal weblog (http://www.aetherial.net/).

The risk of mixing free and non-free

In a world where anything (even SCO…) is possible, we have to remember that risks are taken when free and non-free software are mixed

Edward Macnaghten

If you are reading this you probably use free software. In fact, I think the probability that you are reading this article using free software is extremely high, whether it is "free-as-in-speech" as in Firefox, or "free-as-in-beer" as in Internet Explorer. Free software is increasingly being taken for granted and is almost treated as some kind of legal right in some quarters. And so it should be. A lot of people have had to put a great deal of effort into ensuring sufficient "free-as-in-speech" software exists for this to be so. However, over-basking in the sunshine of carefree internet surfing may be as risky as not using sunblock when going out in the sun. Drifting into over-indulging in the candy-shop of license free word-processing and spreadsheet-calculating may well cost you the IT equivalent of a jean size. The liberty of being able to use free software is still at risk.

The amazing money-making tollbooth

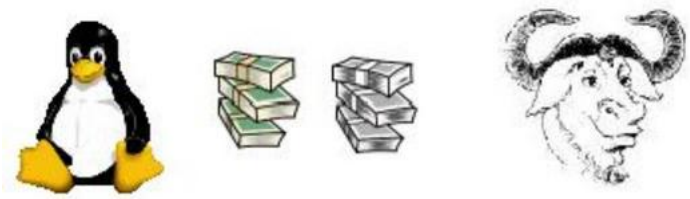
Roads, or the vast majority of them, are free for us to use. Yes - I know - we do pay for them in taxes and other means of government extortion, but with the exception of a relatively small number of toll roads we don't have to pay for them when we use them. If you live in an area where toll roads have become a necessity then please let me apologize for using an inappropriate analogy.

Imagine if you could set up your own tollbooth at the bottom of your driveway though, or if not there, at your nearest major road. You could charge each car a dollar, or even more, as it drove past. Your only cost was the administration of that toll, and you wouldn't have to contribute to the building or upkeep of the road. Imagine that there was no alternative route for the cars to use to get from A to B. Imagine if it was all legal and above board. Wow! You could be a millionaire in no time at all. Forget about lottery tickets - This is the *real* way to get your yacht in the mountains - or whatever takes your fancy.

Fig. 1:

I can imagine cynics saying at this point "imagine if pigs can fly" or "imagine if politicians were honest" etc, and I admit, the above scenario is unlikely, but what if it *could* happen? I doubt if you, or the greater percentage of the population, think this a moral or fair way of earning money, but you would need to be very conscience driven not to fall into the temptation of earning loads of cash that way if you actually could. After all - if you dropped a hundred dollar bill in the street would you actually expect the person who

Fig. 2:



found it to try to give it back? What if you found a hundred dollar bill? Would you take the trouble to trace the owner or hand it to someone with whom you think the owner may enquire it? Taking it is theft - but would that stop you? If you have been in that situation and you did the honourable thing then good for you. I salute you. However, you would be in the minority if you did. The majority of the population would pocket their newfound wealth for an extra treat when they next go out. And why shouldn't they? No one would stop them and honesty is unlikely to be reciprocated if the roles were reversed. In the same vein, why shouldn't you piles of money using inappropriate toll-booths if you can?

> It is very difficult for most people to resist immorally making large amounts of money if they can get away with it

It is very difficult for most people to resist immorally making large amounts of money if they can get away with it. On the other hand, life wouldn't be so grand if someone else could legally put a tollbooth under these terms, say, at the bottom of your driveway. Suddenly this amazing tollbooth is not quite so wonderful after all. It may be unfair on you - but so what? The person doing it would have every right to. If you don't like it: you can just stop using your car. Don't be fooled into thinking that the world is too nice for this to happen, there are a lot of people around who'd be more than happy to take money from you in this way if they could.

I believe, in that case, there would be a race to get as many tollbooths situated at the bottom of as many driveways as possible. It would be a dog-eat-dog scenario, with relatively few participants ending up controlling the vast majority of tollbooths. The effect that it would have would be simply to ensure as many tollbooths inconvenienced and extorted as many citizens in as short a time as possible.

Fortunately for us, there are laws that protect us from that. If any not-so-friendly neighbour tried it, all it would take is one phone call from you and the boys-in-blue would be around to take him away for some serious questioning and possibly a falling-down-the-stairs accident on the way to the cells. You have nothing to fear from a phantom booth mysteriously appearing where it doesn't belong. Your freedom to use the roads is more or less guaranteed. Of course, that's assuming there are sane (or at least not too insane) governments in control of things. There's no danger of your freedom being eroded in this way, or at least, I hope not.

Unfortunately, I cannot say the same about the freedoms of using free software. Government legislation doesn't protect the freedoms of free software, as much as other aspects of life, due to the fact it's considered unimportant and isn't really understood.

How to take away freedoms in software

I'm a software developer. I have developed, and I am developing, free software as specified by the GNU General Public License. I rely on the copyright laws of countries to ensure my software is not used inappropriately and to give me recourse if it is. I have also in my time developed proprietary software, and the company I worked for relied on the same laws to ensure income from license revenue. This is all well and good and, like most people I know, I don't have a problem with this... so far.

It's possible that I could have developed proprietary software that runs on Linux. In fact, I believe that some of the UNIX proprietary code I wrote has now been ported to it. Suddenly we have the situation where, although you didn't have to pay for the operating system, and certain freedoms regarding it are guaranteed, you need to pay a license to run the application on it. This is also fine. It is an important part of the GNU/Linux model. All benefit from it.

However, mixing proprietary code in free systems can have a practical danger.

Going back to the tollbooth analogy imagine a hundred mile long road that goes from one town to another, and has no turn offs or means of entering or exiting it. Now imagine ninety-nine miles of that road being entirely free to use, and one mile of it is privately owned, and you have to pay just for travelling on that one mile. The fact that the vast majority of that road is on public land and is free to use is now

irrelevant. You still need to pay to use the road. None of it, in effect, is available to you, to make use of, without paying. By the same token: let's say I produce a GNU/Linux box, for a small company, with my proprietary application on it. The terms of my license specifies that if any software is installed, modified or altered on a machine with my application on it, or a machine on the same LAN, then the user loses all rights to use my software without reimbursement or recourse of any kind. This would be legally OK for me to do. I'm not violating the GPL because there is no GPL code in my application, and this is where I am putting the restriction. In addition, I'm not putting restrictions on their use of the GNU/Linux programs themselves, so I am not breaking the license there either.

If I place a restriction on one small part of your computer that is vital to you, then I am in effect restricted you in the use of all of it regardless of the freedoms others may have granted you

If I place a restriction on one small part of your computer that is vital to you, then I have in effect restricted your use of the machine as a whole, regardless of the freedoms others may have granted you. Big deal you might suggest - they can dump my application and carry on using GNU/Linux - or run other boxes, not connected with the free software on it. However, this may not be a choice. The expense of dumping the software, if it is a custom-built application, which has been developed over a long time, may well be prohibitive. As for running a separate network or box, that may be unrealistic too. It could well mean two screens on every desk instead of one or some employees not having access to all of the systems. This could also be unacceptable.

Should the above scenario occur then, in effect, I would have succeeded in "un-freeing" the GNU/Linux system for that company. Many of the freedoms the GPL is meant to have guaranteed its users would have been denied them. Furthermore, I would have done it whilst not violating the GPL myself. I would have succeeded in placing a toll on a mile of road but in the process gained control of a hundred.

I realise the above is rare, but occur it does. I know of scenarios where this has happened in the past. Now, with the serious proliferation of software-method patents, it could

Fig. 3:

become easier and easier for such companies to gain control of crucial aspects of a computer's application processing, such that it lets them gain a foothold in the entire computer's works. The SCO group have even tried it directly with GNU/Linux - they never claimed all the code was theirs, just sufficient amounts for it to warrant a per-machine payment for the use of it. Fortunately, it is looking increasingly likely that their lawsuit will fall completely on its face and be buried in the mud where it belongs. However, it may have been a different story if they had their act together better in the beginning, or "if it wasn't for those pesky volunteers at Groklaw" exposing it for the sham it was. It is wrong to be complacent about such things.

It can be hard to resist improperly earning money from free software developers' efforts. There are many who would do anything they could to achieve this

It can be hard to resist improperly earning money from free software developers' efforts. There are many who would do anything they could to achieve this.

Take a look at Sun's Java Desktop System. It's actually SuSE Linux running Gnome with all that goes with it, and Star-Office and some other Sun proprietary software on it. Sun charge you for the entire desktop, and support it. If you install software, not approved by them, on the system: you may find that Sun no longer guarantee uptime and support for it, until you un-install the software. However, either way, you'd still be liable for the annual payments. The fact you could get SuSE, Gnome and even OpenOffice from elsewhere could well be irrelevant in this case - your business

may depend on an aspect of the Java Desktop that is entirely sun's intellectual property. In effect - the freedoms granted to you by the GPL have potentially been destroyed.

Conclusion

Allowing proprietary software to run inside free systems is a necessity in the real world. However, don't be fooled into thinking it's without danger. There are those around you who would use any tactic to obtain revenue from GNU/Linux licensing, or from any other free-as-in-speech software. Placing proprietary and highly controlled applications on systems could be a "foot-in-the-door" first step toward achieving this end. Computers are useless if you can't have practical use of them. If these proprietary applications mean, in effect, that you cannot use any part of the computer - including the free parts - without the permission of the vendor then the "freeness" of the software has been removed. It can happen. When you see proprietary software on a free system - look more closely. Be aware, and beware!

Copyright information

© 2005 by Edward Macnaghten

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at http://www.gnu.org/copyleft/fdl.html

About the author

Edward Macnaghten has been a professional programmer, analyst and consultant for more than 20 years. His experiences include manufacturing commercially based software, in a variety of technical environments, for a number of industries in Europe, Asia and the USA. He is currently running an IT consultancy specialising in free software solutions based in Cambridge UK.

ANSWER SQUAD.com

Tech Questions? We wrote the book!

Feel like roadkill on the Information Superhighway?

Questions about your computer, the Internet, the program you're writing, or even just the commercial application you're stuck working with every day? You've tried searching the Internet for solutions, with mixed results. You've asked people on various mailing lists, just to be flamed or answered with "rtfm!" You've tried tech support at Microsoft, Apple, Adobe or Macromedia, just to be overwhelmed by their cost. And here you are, still puzzled, still stuck, and still having to endure your computing environment rather than enjoy it.

Get roadside assistance with the AnswerSquad!

We're a group of best-selling authors and tech experts who are creating new, innovative and more efficient ways to help you find your computer nirvana: having everything work the way you want, when you want, and how you want.

Visit our website to learn how you can get the answers you need now.

AnswerSquad.com

Center of the System Administration

www.sbin.org

Open look at common things

Go Open

Open Source Software, which originally developed out of the passion of hackers and geeks, has now become the basis of real business solutions. Low TCO, high reliability, security and performance of Open Source software make the shift from "closed" technologies and standards to modern, open solutions inevitable.

CSA offers development, deployment and support services based on Open Source technologies and standards.
Up-to-date solutions for small and medium size businesses are available for you today with guaranteed full 24/7 support and no license fees.

Today, being closed means being left behind

+7 8793 365584

Designed by ITA "InfoMaster"

When free meets proprietary

Three free applications on proprietary systems, three proprietary applications on free systems

Daniel O. Escasa

In a dream world, all software would be free. However, we spend enough time with our eyes open to realize that some situations call for proprietary software, either as a desktop or as a server application, on a free system. On the other hand, those stuck with a proprietary operating system can still enjoy free software applications. This article will list a few situations where free software and proprietary software can mix, and give three examples of each.

Why free and proprietary can mix

Examine this scenario: your company is moving your servers over to a free system. The trouble is, you have an Oracle database that's been running for x years and it's tweaked so perfectly you can't afford the time and effort to scrub it and migrate to a free relational database management system (RDBMS). Rest easy: Oracle is available on free systems, albeit certified only on certain distributions.

Another scenario: your company policy is free systems on desktops, with a few selected exceptions. You're one of the poor sods who pass the Free-OS exemption test because of some esoteric application not available on free systems. However, your company can't afford a license for MS Office, and you do need a Microsoft file-compatible word processor, presentation-authoring tool, or spreadsheet. OpenOffice.org to the rescue.

Situations such as those above are not limited to corporate desktops. Your home PC might have an important proprietary application that doesn't have any free counterpart. You can't resort to dual-boot because that just takes too much time. So, you can run an emulator on a free system to run that proprietary application. In an extreme case – e.g., your software won't run in an emulator – you might be forced into Windows but choose to run mostly free applications.

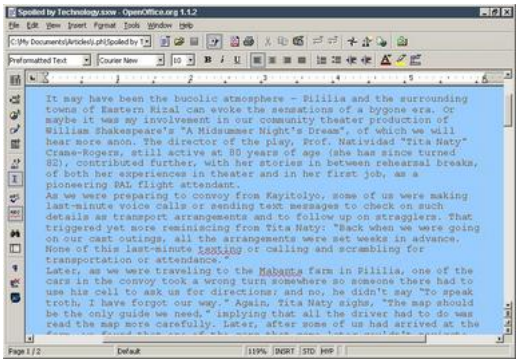
For this article's purposes, treatment of proprietary systems will be limited to Windows 9x/ME/NT/2K because they're the dominant platforms. Other proprietary systems are OS/2, BeOS, and MacOS X, and most free software applications for Windows are available for them as well. I will define free systems as GNU/Linux and the BSDs and I will refer to them as "Freenix" for convenience.

Free software on proprietary systems

The OpenOffice.org suite and the Firefox Web browser are the most prominent free software applications on proprietary systems. In addition, server applications such as Apache, MySQL, and PostgreSQL are also available for Windows.

OpenOffice.org (`http://www.openoffice.org/`) began life as StarOffice, an office suite by a German company named Star Division. StarOffice was available for Windows, OS/2 (up to version 5.2), and Unix. Sun Microsystems bought StarOffice in August 1999 and subsequently opened up the source code to the community. The result was OpenOffice.org, now at version 1.1.4 for all available platforms – which include Windows, OS/2 (for a

Fig. 1: OpenOffice.org Writer



fee), and Freenix. OpenOffice.org includes a full-featured word processor, presentation-authoring tool, spreadsheet, and drawing tool. It is nearly 100% file-compatible with MS Office.

As an aside, StarOffice, now at version 7, is still available for Windows, GNU/Linux, and Solaris, for a suggested retail price of US$79.95. However, academic use is free except for shipping and handling. In addition, Sun also offers site licenses. Lastly, there's an electronic download available for US$59.95. StarOffice has all the components of OpenOffice.org, and adds a simple database manager. The Sun FAQ claims that SO and OpenOffice.org share the same code.

The Mozilla suite includes a Web browser, chat client, newsgroup reader, POP/IMAP client, HTML composer, and the Jabberzilla multi-protocol IM client

Firefox is a child of the Mozilla (`http://www.mozilla.org/`) project which also produced its bigger brother the Mozilla suite. The latter includes a Web browser, chat client, newsgroup reader, POP/IMAP client, HTML composer, and the Jabberzilla multi-protocol instant messaging (IM) client. Firefox is a stand-alone browser, now at version 1.0, available for the same platforms as Mozilla and OpenOffice.org. Like the advertising-sponsored Opera and the Mozilla browser, Firefox employs tabbed browsing. This means that you can browse several pages in different panes within the same window, and switch from one pane to another by clicking on the tabs at the top of the panes.

Fig. 2: Firefox in full-screen mode. Note the tabs at the top of the viewing area

The most exciting feature for me is the use of the XML User-interface Language (XUL, pronounced zool to rhyme with 'cool') to create extensions, which as the name implies, extend the features of Firefox. The one I find most useful is WebmailCompose, which converts a mailto: link to a link to my selected Webmail account's compose window. In other words, when I click on a mailto: link, a new tab opens up where I can compose a mail message from gmail. Also, here's an interesting sample XUL application (`http://www.faser.net/mab/chrome/content/mab.xul`) that should give you some idea of XUL's capabilities. A real-world application is the Wizz RSS News Reader, which uses the sidebar as the control panel. I personally find XUL interesting because it looks to be one of the building blocks in creating a class of applications known as Rich Internet Applications (RIAs), which use the browser as a universal front end.

The Web server we know as Apache, officially the Apache HTTP server, is the most famous output of the Apache Software Foundation (`http://www.apache.org/`). Apache serves approximately 67% of the world's websites. It is now at version 2, available both for Windows and free systems. A 2.1 version is, as of this writing, in beta.

Free desktop software on Windows can make sense, depending on the situation. However, I personally question the usefulness of running free server software on Windows. They run more efficiently on free systems, which in turn can run on older – and therefore, cheaper – hardware. I would only run a free server on Windows for experimention and education not for production.

Proprietary software on free systems

Vendors of proprietary software have made their products available on free systems for one reason or another. Oracle (http://www.oracle.com/), for instance, saw GNU/Linux as an opportunity to expand their market. Some applications – Opera (http://www.opera.com/) and VMWare (http://www.vmware.com/), to name two – were available on free systems from the start, and it was natural for the latter.

Oracle created a stir in the industry when they announced the availability, sometime in 1998, of their database products on GNU/Linux. According to the Oracle-on-Linux FAQ, the following are the certified and supported distributions:

- Red Hat Enterprise Linux AS and ES
- SUSE LINUX Enterprise Server
- Asianux 1.0, which includes Red Flag DC 4.1 Asianux Inside and Miracle Linux 3.0 Asianux Inside
- UnitedLinux 1.0

This is not to say that Oracle won't run on other GNU/Linux distributions, or on the BSDs, just that Oracle can't certify and support all of those systems.

The FAQ adds that "All key Oracle products including Oracle Database 10g with Real Application Clusters, Oracle Application Server 10g, Oracle Collaboration Suite, Oracle Developer Suite 10g, and Oracle E-Business Suite are available for Linux." Those of you who run Oracle know the details of those products better than I do. Oracle distributes these in no-fee 650MByte .gz (gzip'ed) files. Those of you without the bandwidth can order a CD from Oracle for a minimal handling fee. Pricing takes effect only when you deploy a database, and starts at about US$49 per user, and moves into per-CPU rates depending on the application. Oracle has the reputation for being the choice for enterprise-class databases. However, free RDBMSes are encroaching on this space. The two most popular ones are MySQL and PostgreSQL, both of which were mentioned earlier. As an aside, MySQL also has a paid-support license option which – need I remind you? – does not detract from its being free software.

VMWare is a software implementation of a virtual machine – i.e., it enables applications from other operating systems to run on the target machine. Thus, in MS Windows,

Fig. 3: VMWare's web site

VMWare will open up GNU/Linux as a task on your Windows desktop; on the other hand, VMWare will open up Windows as a task on your Freenix X desktop. And, once you have Windows-on-X, you can run most Windows applications. However, graphic-intensive games, such as Duke Nukem or Far Cry, will run far too slowly – if at all – to be playable. Feedback from users indicates that Microsoft Office will run under VMWare, albeit at a slight speed penalty.

VMWare is available as a no-fee download in .tar.gz (tar'ed, then gzip'ed) format. You are free to try it out for 30 days, after which the program dies unless you purchase a license: the workstation edition is US$189 for electronic delivery, US$199 for packaged. Network licenses are also available, starting at US$1,694 for two CPUs.

If the licensing fees are too steep for you, there are several free software options – wine (http://www.winehq.com/)(Wine is Not an Emulator) and bochs (http://bochs.sourceforge.net/). The setup for these may not be as smooth as VMWare's though. I, for one, never got my wine working on my FreeBSD box, although I admit that maybe I didn't try hard enough, since it wasn't a priority. On the other hand, it is free software.

Opera is a no-fee download, and you can continue using it for as long as you want, although you'll have to put up with ads that take up the top portion of the task window. US$39 gets rid of those ads. Opera claims to be the "safest and fastest full-featured Web browser on the market." People who have tried both Firefox and Opera confirm that Opera is faster. However, the consensus was that Firefox adhered more closely to W3C standards than Opera. Besides, Firefox is free software, whereas Opera's source isn't even available to begin with.

Other proprietary software available on Freenix are Sun's

Fig. 4: Opera's web site

StarOffice, Sun's Java, and IBM's Websphere. Take note that, while Sun does make the source to Java available, you aren't free to modify it. As to IBM's Websphere, it's a no-fee download like Oracle, and then you pay for deployment.

Conclusion

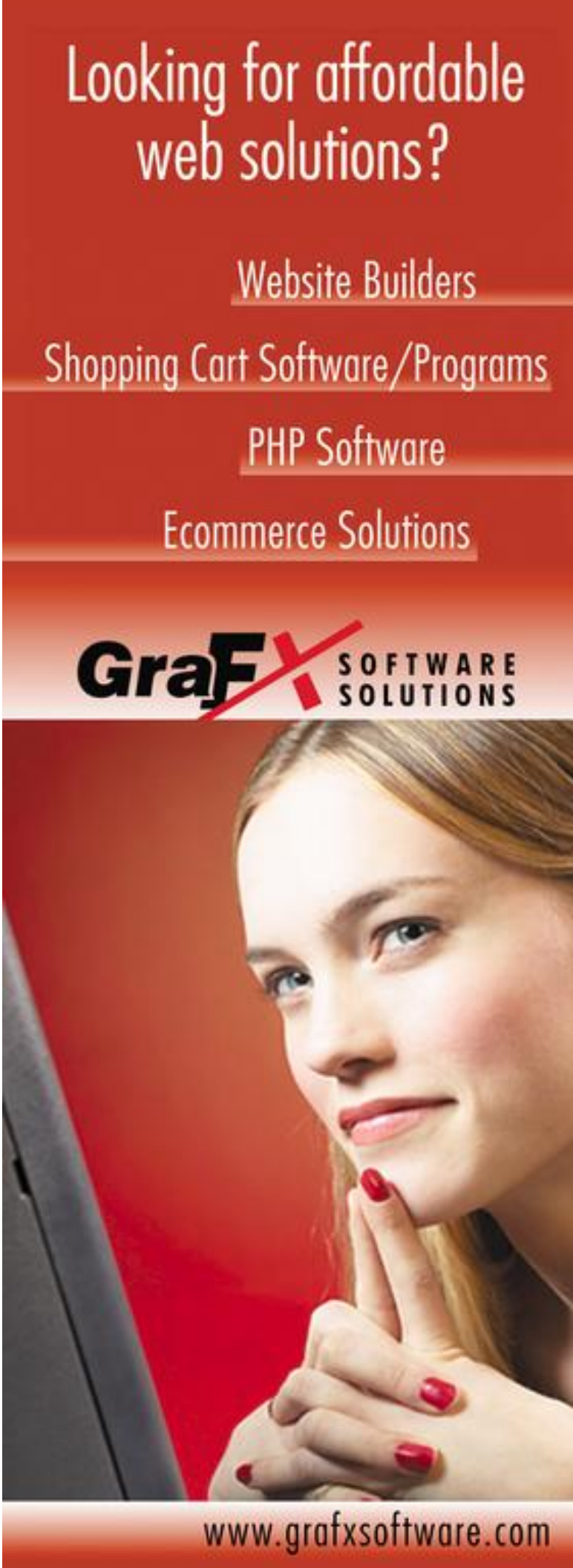
So, if you're stuck in a proprietary system, don't despair: you're still invited to the free software party because there are ports of free software. Conversely, if you're already enjoying free software bliss at the operating system level, don't knock proprietary software.

Copyright information

© 2005 by Daniel O. Escasa
This article is made available under the "Attribution-NonCommercial-ShareAlike" Creative Commons License 2.0 available from http://creativecommons.org/licenses/by-nc-sa/2.0/.

About the author

The author is a freelance writer and consultant with over 20 years employing what others might call "oddball technology". He simply thinks it's technology off the beaten path. More of that, and other miscellany, at his blog (`http://www.descasa.i.ph`).



ALL MUSIC AND ARTWORK IS RELEASED UNDER CREATIVE COMMONS ATTRIBUTION SHARE-ALIKE LICENSE

WWW.LIBREESOCIETY.ORG
WWW.CREATIVECOMMONS.ORG
WWW.WIKIPEDIA.ORG

E-LOCA-TRONICA 2005
STICK TO WHAT YOU DON'T KNOW

HTTP://WWW.LOCARECORDS.COM

INFO@LOCARECORDS.COM

YOU CAN DOWNLOAD MUSIC, VIDEOS AND TEXT BY VISITING WWW.LOCARECORDS.COM

LOCA RECORDS PO BOX 233
BRIGHTON BN2 3WX UK

XML: WYSIWYG to WYSIWYM

A brief look at XML document authoring

Saqib Ali

It all started with cavemen and their cave drawings. All cave drawings were WYSIWYM (What You See is What You Mean). I mean (no pun intended), if you saw a cave drawing, in which a hunter was chasing a mammoth, it meant that a hunter was chasing a mammoth. There were no two ways to interpret the cave drawings. Then came alphabets and words. With words came plain text or documents. Then came XML/SGML for adding information to a document relating to its structure and/or content. An XML document contains both content (words) and an indication of what role the content plays. It's a markup language that adds meaning to the meaning. With the advent of XML, there became a need for XML editors.

Basic Terminology

DTD

Document Type Definition. An XML DTD, written in EBNF, defines the structure and syntax of an XML document. Essentially, it defines the elements, entities, and the content model. For more info, please see this link (`http://www.xml-dev.com/blog/#64`).

Open Standard DTD

Open Standard DTDs are Document Type Definitions that are publicly available for use with various XML aware applications. An example of an Open Standard DTD is DocBook XML DTD. In contrast, WordML developed by Microsoft, is a proprietary XML DTD. WordML needs to be licensed, in order to be utilized by non-Microsoft applications.

XML

XML (eXtensible Markup Language) is a meta markup language that can be used to create other markup languages. XHTML is an example of markup language created using XML.

XML Schema

Similar to DTD, except it's written in XML, and is also capable of defining data types and putting constraints on the content. For more info, please see this link (`http://www.xml-dev.com/blog/#64`).

XSLT

XSLT is a template, which defines what transformations need to be performed on an XML document. XSLT allows XML data to be shared among various XML aware applications, which don't necessarily use the same XML schema. In the context of document authoring and content creation, an XSLT defines what the formatted output will look like.

Valid

An XML document that conforms to rules as defined by an XML DTD or schema is valid.

Well-formed

An XML document that conforms to the syntax rules of XML is well formed.

Proprietary file formats

The popular way of authoring a document is to use WYSIWYG tools (e.g. MS Word or Frame Maker). Documents created using these applications only have formatting information, but lack semantic information (i.e. information related to the structure of the information contained within). Since the formatting information is proprietary to each vendor and the file format is binary, documents created using these applications are locked into the file format of the application's vendor. If you wish to switch word processors (e.g. Microsoft Word to Corel's WordPerfect), you have to go through the painful process of converting all of your documents to the new proprietary format, and often lose valuable data during the conversion process.

Enter XML based document authoring

In contrast, XML allows a document author to create content in a "presentation neutral" format that captures the semantics of the content, rather than the presentational format. XML introduced a way of authoring documents that is completely free from the limitations of old fashioned, non-interchangeable, binary file formats (e.g. DOC, XLS etc). Open Standard XML Document Type Definitions (DTDs) and XML Schemas allow the document authors to create consistent, structured documents regardless of the XML editor they are using. A true XML editor supports any XML DTDs or XML Schema. Thus, the document author is able to move from one XML editor to another, without the fear of the content being locked into one proprietary file format. This is good for the document author, but not good for the software vendor who developed the editor. The only way for vendors to compete for customers is to offer editors that provide the more editing features. If they don't, the customer will just move to a different editor, without losing any content. This has paved the way for the XML editor wars, where each vendor wants to develop an editor that has better features than its competitors' editors have.

Features offered by XML editors

Despite the fact that XML is the Holy Grail for publishers and document authors, nobody wants to use Vi or Notepad to create an XML document. They want tools that'll make XML editing easy and painless. So, the paradigm of WYSIWYG and WYSIWYM was introduced to the XML world. Essentially the document authors want XML editors that can:

1. Highlight syntax errors
2. Automatically complete XML Tags
3. Indent properly
4. Check for validity against an XML Schema or a DTD
5. Check for XML well-formedness
6. Allow viewing and editing of XML documents in a tree view, and
7. Fix the kitchen sink

Various types of XML editors

The positive outcome of the XML editor wars is that we currently have large number of very good and feature-rich XML editors to choose from. These editors can be divided into three categories: WYSIWYM, WYSIWYG and Text-Based.

WYSIWYM

What You See Is What You Mean is a paradigm that is related to the XML editors, which accurately display the information that is trying to be conveyed, rather than the actual formatting. Since XML doesn't define the actual formatting of the content, these editors are very useful in visually creating and managing the data. A WYSIWYM editor allows a document author to edit an XML document by interacting with a feedback text, generated by the editor. This presents both the knowledge already defined and the options (tags, elements, attributes, etc.) for extending and modifying it. Thus, a WYSIWYM XML editor alleviates the need for the document author to memorize all of the tags, elements and attributes of XML DTD or Schema. WYSIWYM XML editors are also known as *semantical editors*.

Butterfly XML is a powerful WYSIWYM editor, freely available under GNU Public License. In addition to the features mentioned above, it is capable of automatically generating an XML Schema or a DTD based on XML file.

WYSIWYG

What You See Is What You Get is a paradigm that is related to word processing and publishing. It refers to editing

Fig. 1: XML Editors

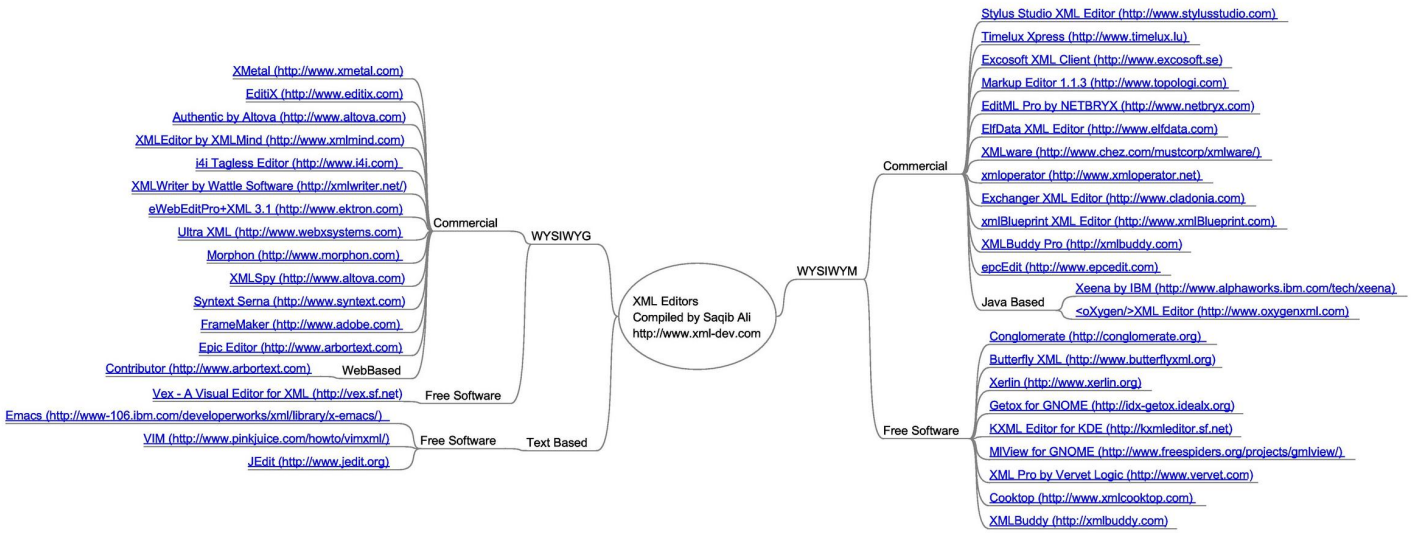


Fig. 2: Butterfly XML is capable of presenting the elements that are available at the location of the cursor as defined by the DTD or the Schema

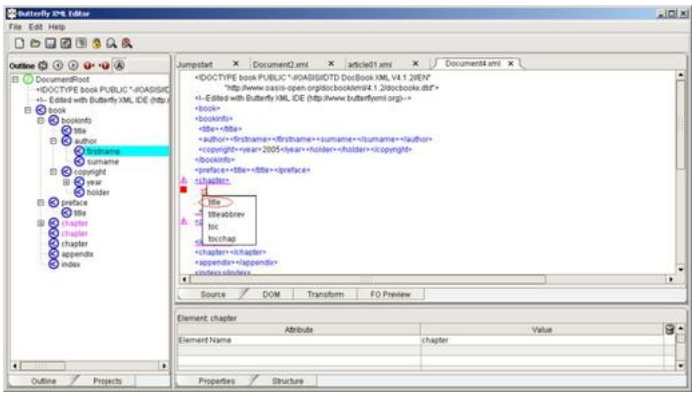
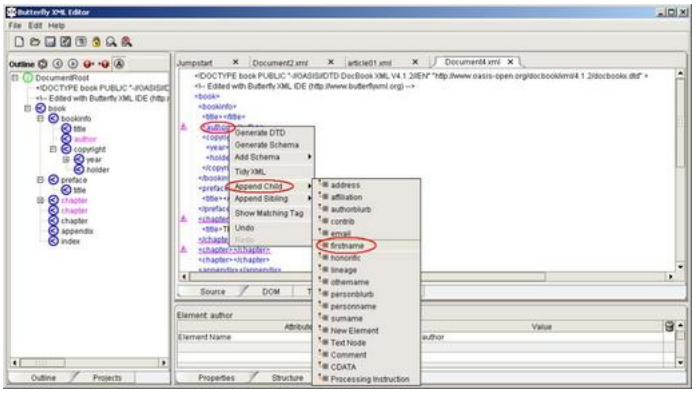


Fig. 3: Right clicking on any tag will display all the child and sibling tags that are available as defined in the XML Schema or DTD.



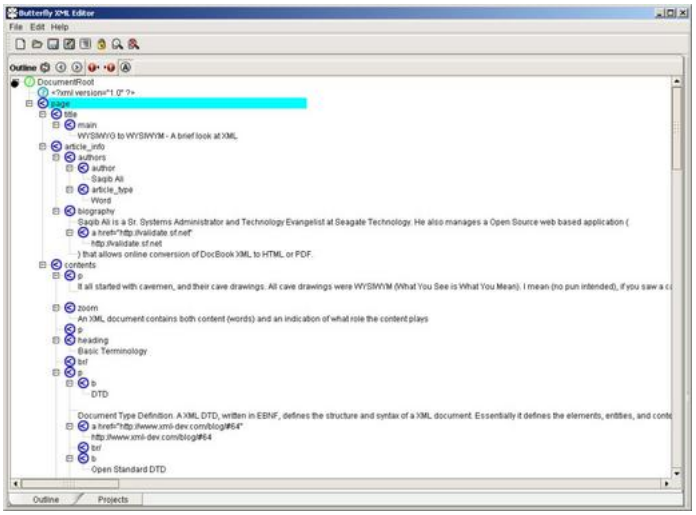
software that makes sure that the image seen on the screen closely corresponds to the final formatted output. That does *not* mean that WYSIWYG XML editors exclude the semantics of the XML content. All it means is that a WYSIWYG XML editor allows the document author to edit the XML content with the XSLT applied to the content. So the document author doesn't see the XML elements and attributes, instead he/she sees the formatted output as defined by the XSLT.

Vex (Visual Editor XML) is a very capable WYSIWYG editor that uses CSS to provide authors with an MS Word like user interface to create and edit document. It is well suited for "document-style" documents such as DocBook and XHTML, but is not the best choice for data collection.

Text Based

They are most rudimentary type of XML editors, where the document's author has to manually type in all of the tags and attributes. However, these editors usually check for good structure and sometimes for validity as well. Both Vi and Emacs can be configured to act as a powerful text-based XML editor. The following documentation can be used to configure any of these free editors so they can be used as an XML editor:

Fig. 4: ButterflyXML offers a Tree View of the XML document. A tree view helps the author to better visualize the semantics of the content.



- Using Emacs for XML documents (`http://www-106.ibm.com/developerworks/xml/library/x-emacs/`)
- Using Vim as XML Editor (`http://www.pinkjuice.com/howto/vimxml/`)

Fig. 5: Like ButterflyXML, Vex is also capable of listing the elements that are valid and inserting them at the current caret location. Available attributes for each element are displayed as well.

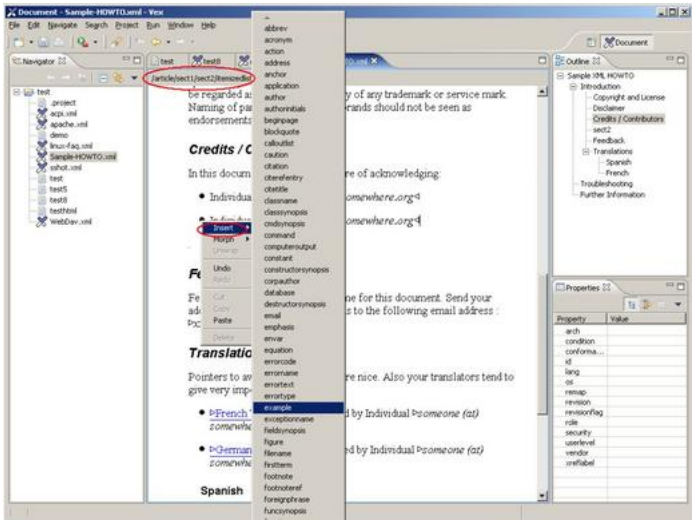
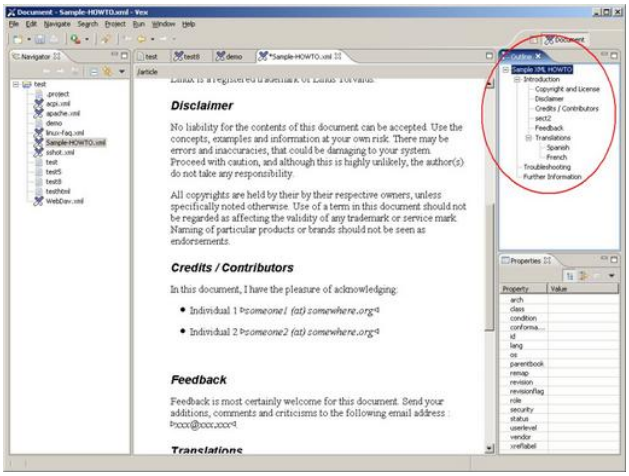


Fig. 6: Unlike ButterflyXML, Vex does NOT provide a tree view of the whole XML document, however it does provide an outline of the document. Outlines provide a good, high-level view of the semantics of the document.



Copyright information

© 2005 by Saqib Ali

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at http://www.gnu.org/copyleft/fdl.html

XML allows a document author to create content in a presentation-neutral form that captures the semantics of the content, rather than the presentational format

About the author

Saqib Ali is a Snr. Systems Administrator and Technology Evangelist at Seagate Technology. He also manages a free software web based application (`http://validate.sf.net`) that allows online conversion of DocBook XML to HTML or PDF. Saqib is also a active contributor to The Linux Documentation Project

A laptop, a coffee, and disaster recovery

Why you should have an effective backup strategy

John Locke

Last week, my laptop died a sudden spectacular death-by-drowning, as a full cup of coffee poured into its keyboard. It emitted a pop sound, and the screen and the power shut off.

What would your reaction be? Mine was to immediately unplug the power cord and remove the battery. Then I took it over to the sink and poured out the coffee. Remembering tales of people flushing keyboards with water, I ran some fresh water over the keys and then set to work. I removed the keyboard, the palm rest, a few of the inner cards, and let it sit without power for several hours. Apparently, not long enough.

Later that day, anxious to find out whether it was really dead or just comatose, I plugged it back in, crossed my fingers, and pressed the power button. The power light came on, I heard the fan start, and for a second or two, I was hopeful. But then... another pop, and it was dead. No further cleaning, drying, or care could resuscitate it over the next few days, so it's currently back at the IBM factory going through open-heart surgery, if not a total replacement.

What can go wrong?

Obviously, if I didn't have a good backup of my data, such an event could have been catastrophic to my business. As it was, the loss of my laptop was merely an expensive hassle. Actual events such as this one can provide a good reality check for your disaster recovery strategies.

Many things can happen that have a similarly disasterous effect. Being prepared for disasters can make life easier if the event actually occurs. Let's take a quick look at some possible computer disasters:

- Coffee fries the laptop.
- A service technician copies sensitive data from your computer.
- Your laptop is stolen.
- Your data is erased by a malfunctioning airport X-ray machine.
- Your house burns down.
- Your PDA is stolen.
- A virus infects your files, and your recent backups.

Which of these disasters could make you go out of business? Or subject you to identity theft? Or to a lawsuit from your customers for leaking their information?

Many things can happen that have a similarly disasterous effect. Being prepared for disasters can make life easier if the event actually occurs

Several issues come into play here, and you need to consider all of them:

1. Loss of use of your equipment.
2. Compromise of sensitive data.

3. Loss of data, including corrupt backups, deleted files, or important files you change accidentally.
4. The overall risk of any of these events.

These are some of the considerations to be made when planning your disaster recovery strategy, and I'll get into more detail in future issues.

The best backup system is one you don't have to think about at all. Ideally, your laptop or workstation should have no data that's not also stored somewhere else

How do I back up my data?

There are several approaches to doing backups. It used to be that most places would get a tape-drive and put entire system backups on a schedule to run overnight. Tapes aren't always reliable, though, and it can be very difficult to retrieve individual files.

These days there are a lot more options. Writable CDs and DVDs provide a cheap way to create permanent backups, and create historical archives. USB thumb drives make for a simple way of instantly making a copy of important files. Large hard drives are cheap to buy, and when installed in an external drive enclosure or removable casing, become the most economical way to back up large sets of data.

If you have a Local Area Network (LAN), you can backup important files by simply copying them to other computers. If you're in an office environment, I highly recommend having a server, and concentrating your backup efforts onto the server instead of individual workstations.

The best backup system is one you don't have to think about at all. Ideally, your laptop or workstation should have no data that's not also stored somewhere else. We're all human afterall and the more work it is to do a backup, the less likely you are to have done one when you need it. So how did I do, when my laptop died?

Painful self-analysis

I probably rely on my laptop for a wider variety of data than most people. An important part of your backup strategy is identifying what data needs to be backed up, how often, and where. Here's a list of what was on my computer:

- Email
- Contact addresses
- Calendar/Schedule
- Password database
- Sensitive customer files
- Sensitive business data
- Finances
- Non-sensitive customer files
- Non-sensitive business documents
- Prototype databases for web application projects
- Prototype web applications under development
- The operating system and application software
- Configuration details about the machine itself

This list is probably a lot longer than yours, but it covers most of the different types of data used by the majority of small businesses, and may include a couple things you haven't thought about. So, how did I go with my disaster recovery strategy:

Email

A perfect score. My email system uses IMAP, which keeps all mail on the server. Every single email in my mailbox is intact on the server, and immediately available from any other computer.

Contact addresses

Not so good. My contact addresses are currently stored in Evolution, my main email software. I periodically back up by synchronizing to my Palm, but I had been having problems with the synchronization, so I hadn't done that for a couple weeks. I could have lost the last two weeks of any changes to my address book.

Calendar and schedules

Ouch! My strategy was completely inadequate in this area. Evolution has great calendaring capabilities and I have several different calendars set up in Evolution, each for a particular type of work. My meeting calendar gets synchronized to my Palm PDA. My social and meeting calendars are merged to publish free/busy information so others can schedule meetings with me. I then keep several other calendars to block out time for different tasks: billing, taxes, security updates, client projects, and articles. In Evolution I can easily show or hide different calendars, and see when

I'm available for a meeting. But the downside is, these calendars are all stored only on my desktop. Evolution does not yet support storing calendars on a server - though you can read remote calendars. Aside from the meeting calendar, the remainder of my tasks and work schedule would have been completely lost.

These types of events always serve as a wakeup call, highlighting areas for improvement

Password database

Okay. It's there, a backed up copy from my PDA. It's also encrypted, though the encryption is not as strong as other encryption methods. At least it'd take substantial effort to crack, and I can trust IBM, with its reputation, not to mess with it.

Sensitive customer and business files

Good. Not only are all my customer files in a server-based document repository, available for instant access from any other computer, but the copies on my laptop are on a strongly encrypted partition. My clients and I have nothing to worry about when the hard drive goes into the service depot: the encryption method used is the Advanced Encryption Standard (AES), which is the US Government standard for strong encryption and currently has no known flaws.

Finances

Super! My finances aren't even stored on this machine, even though it's my primary workstation. I use a web-based application, with an encrypted connection. My business finances never leave my server, and the server is backed up nightly.

Non-sensitive customer files, prototype web applications

Good. Again, all customer files are stored in a document repository, and I work on a local copy. Whenever I'm done for the day, I check these files back into the server. I could have lost a few paragraphs, written in a new proposal that morning; but in this case, there was no catastrophic loss.

Non-sensitive business documents

Poor. I had a lot of marketing material in my computer that was not backed up anywhere else. Chunks of text getting prepared for the web site, a couple of time and mileage tracking spreadsheets, a few other assorted files were not properly added to my document repository. The mileage tracking spreadsheet in particular would have been a tough loss.

Prototype databases

OK. If you're not a developer of some kind, you probably don't have any databases running on your laptop. I have three: MySQL, PostgreSQL, and Oracle. It's one of the main reasons I got a new laptop this year - to get the performance I need to handle the types of development I'm doing. Databases often can't be backed up as files, especially when they're running. But there's nothing in the databases on my laptop beyond sample data, and the structure of each database is regularly exported to a file and added to my document repository. The document repository itself is a type of database that entails special backup procedures, but that's not on my laptop, so that's a story for another day.

While I had most of my critical data automatically backed up, there was a significant hassle involved in losing the data stored only on the laptop

Operating system and software

Just fine. Many people back up the entire hard drive, just because that's what they think is necessary. I never back up the operating system or installed programs. In the Linux world, it's almost always better to install the newest version of any software - catastrophic loss of a computer is just a great opportunity to upgrade. In the Windows world, you tend to need to periodically reinstall the entire operating system periodically anyway - especially the way recent spyware and viruses have infiltrated so many machines. Much better to start fresh and install the programs you need when you need them, either from new versions or directly from the installation CDs that you have in the bottom of that drawer.

System configuration

Total loss. On one hand, I try to keep the system configuration as close to the default as possible. But I had done a fair amount of custom configuration on this laptop to support things like the infrared port, the modem, and the wireless card, as these things don't always work reliably out of the box in Linux. I have notes around here somewhere, but didn't put them in a central place and some of the stacks

of paper around here qualify as their own disaster needing recovery.

Result: needs improvement

While I had most of my critical data automatically backed up, there was a significant hassle involved in losing the data stored only on the laptop. In particular, I need to improve my contact, calendar, and a few less critical business documents. I also need better documentation of my system configuration. This would help me get this laptop back up, and running much more quickly.

Fortunately, I didn't lose a thing. I went down to my local electronics store and bought a little drive enclosure for a laptop hard drive, plugged it in and recovered everything on the drive, including the things I didn't have well backed up. All the same, these types of events always serve as a wakeup call, highlighting areas for improvement. Top of my list, based solely on the value of the data, is getting a better system in place for storing my contacts.

How would you fare if you lost your primary computer?

If you don't feel confident in your current backup strategy, we at Freelock Computing would be happy to help you figure out a better plan. Next month I'll look at options for encrypting sensitive documents. It's easier than you may think.

Copyright information

© 2005 by John Locke
(The following license is effective immediately)
Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

About the author

John Locke is the author of the book *Open Source Solutions for Small Business Problems.* He provides technology, strategy and free software implementations for small and growing businesses in the Pacific Northwest through his business, Freelock Computing (`http://freelock.com`).

A PLANT NEEDS WATER TO GROW
FREE SOFTWARE MAGAZINE
NEEDS SUBSCRIBERS TO PROSPER!
Free Software
MAGAZINE
BY SUBSCRIBING YOU WILL BE SUPPORTING A MAGAZINE WHICH BELIEVES IN FREE SOFTWARE. ALL OUR ARTICLES ARE RELEASED UNDER THE GNU FREE DOCUMENTATION LICENSE, ENHANCING EXISTING INFORMATION ON FREE SOFTWARE.
SUBSCRIBE NOW!
HTTP://WWW.FREESOFTWAREMAGAZINE.COM/SUBSCRIBE

The importance of LDAP

The past, present and future of a battered protocol

Tom Jackiewicz

All that you know about Lightweight Directory Access Protocol (LDAP) is wrong. From its inception to perceived usefulness, and ultimately, until the marketing department got a hold of it, LDAP has grown. It started as a useful protocol and a data structuring methodology (known by only a few), and became the latest and greatest way to synergize your *action items* and find parity with your eMarketing growth plan.

The freedom given to those who choose to deploy LDAP has ultimately led to problems in interoperability

What does this mean? Hopefully your answer is the same as mine - nothing. The importance and growth of LDAP should be based on how the protocol has adapted in the past, and how we keep on innovating and adapting it as technology grows. Unfortunately for us, the marketing department has already taken a hold of LDAP, eaten it up, and - if we are not careful - it will spit it out.

The origins of LDAP

LDAP was first implemented at the University of Michigan in 1992 as a way of creating an interface to DAP over TCP/IP. This eventually evolved into a stand-alone system utilizing data structures based on types stemming from X.500. Over time, the popularity of OSI waned and TCP/IP became the de facto standard accepted for networking. One of the reasons for the failure of OSI (and its directory solution – DAP) was the complexity of the data definition and the infrastructure required to use it. No flexibility was given in any of the OSI standards and it became extremely cumbersome to deploy.

With OSI's failure and TCP/IP's acceptance, the LDAP "community" (which at the time consisted of college age engineers meeting over pizza and beer) became more ambitious and tried to invade the space of directories and started to innovate instead of just following X.500's lead. Following the original RFC's for X.500 and LDAP, it can be seen that while X.500 was trying to solve problems that didn't exist (i.e. how to replace yp, how to create a more structured information standard that was even more painful to implement), LDAP was clarifying how to access information and proposing formats for URL standardization and search filters.

Fig. 1: The OpenLDAP web site

Because of its simplicity, ease of implementation, availability, and a little bit of luck, LDAP became a good way of centralizing information. LDAP provided an easy solution where synchronization scripts and custom code were used to keep information, on large systems, consistent. Within a relatively short time, email systems were relying on LDAP as the single authoritative source. Authentication systems were no longer relying on their own customized solutions and flat files. Netscape looked to LDAP to provide the central repository for their initial suite of products.

Following the original RFC's for X.500 and LDAP, it can be seen that while X.500 was trying to solve problems that didn't exist, LDAP was clarifying how to access information and proposing formats for URL standardization and search filters

All of a sudden, LDAP was the quick and easy solution to many of the problems that system administrators faced. The community, large and growing, was updating standards in order to help solve the problems that the users were facing. Once other vendors were replacing their back-end databases with LDAP, it was obvious that this little experiment had finally gained acceptance.

If it ain't broke, don't fix it

One can easily conclude that the success of LDAP can be attributed to how fast technology evolved during the latter part of the 20th century and the general laziness of the information technology community. While new products, solutions, ideas and toys where becoming available, it was too difficult (and too expensive) to implement a database (such as Oracle) for storing profile information for all of a system's users. It was equally frustrating, migrating flat files across different systems, which required similar information. Deploying an LDAP directory didn't require much in the way of an investment of time. It could also be used by some of the new products that were being released. One LDAP directory provided the same authentication and profile information for all of the new toys that system administrators wanted to experiment with. So if it didn't live up to their expectations, only hours were wasted (instead of the resources required to deploy an instance of Oracle).

Unfortunately, in today's fast-paced world, a product that isn't constantly updated with all the new bells and whistles is not trendy or exciting. Vendors often remove features just to add them again a few revisions later. The LDAP community, through RFC's, fell into this trap and started solving problems that didn't exist. RFC's were proposed to adapt DNS information into LDAP. Even complicated schema was proposed to store Java objects. LDAP was slowly moving into areas that it didn't need to exist, and that just made it more complicated. These were the same types of endeavours that destroyed OSI, X.500 and DAP. Those who fail to learn from past mistakes are destined to repeat them.

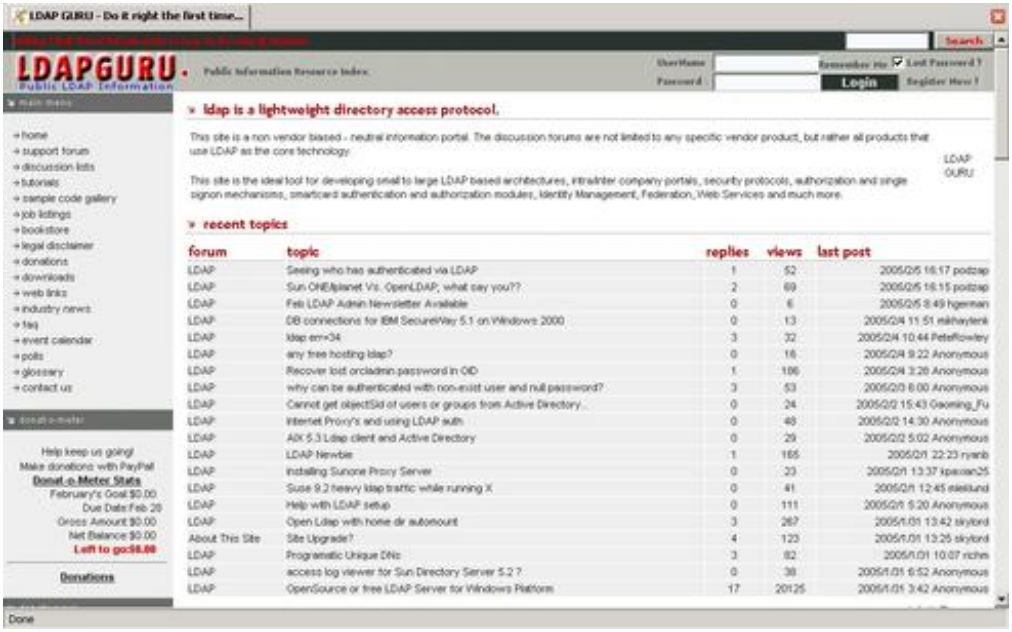
Vendor interpretation

The acceptance of a technology by a vendor is a blessing for many. For others, it can be a fast spiral towards obscurity. Initially, LDAP gained widespread acceptance by the information technology community because of its use within Netscape's suite of products. However, as other vendors started to tap into the possibility of LDAP, their proprietary system background began to invade LDAP's space. LDAP gained popularity because of well established standards and the ability to be protocol dependent and vendor (or implementation) independent. The data stored in the original University of Michigan LDAP server could be exported and put into OpenLDAP with ease. However, vendors (like Sun, Novell, and Microsoft) chose to implement good (but proprietary) features that required only the use of their implementation. Direct calls for authentication and authorization via LDAP became requirements of proprietary plug-ins and new integration layers. While some of these features went beyond the scope of LDAP, standards should have been written, and these new feature sets could have easily become part of the standard LDAP feature set. Instead, the question quickly went from "Are you using LDAP?" to "Which LDAP are you using?".

To make things worse, the way vendors added LDAP to their offerings was questionable.

Early adapters integrated all of their product offerings with LDAP - despite their dependence on proprietary features of their LDAP implementations.

Fig. 2: The site LDAPGURU.COM is helping to provide the right LDAP information


LDAP GURU - Do it right the first time...
LDAPGURU.
Public LDAP Information
Login
» ldap is a lightweight directory access protocol.
This site is a non vendor biased - neutral information portal. The discussion forums are not limited to any specific vendor product, but rather all products that use LDAP as the core technology.
This site is the ideal tool for developing small to large LDAP based architectures, intra/inter company portals, security protocols, authorization and single signon mechanisms, smartcard authentication and authorization modules, Identity Management, Federation, Web Services and much more.
» recent topics
forum
topic
replies
views
last post
home
support forum
discussion lists
tutorials
sample code gallery
job listings
bookstore
legal disclaimer
donations
downloads
web links
industry news
faq
event calendar
polls
glossary
contact us
Donations
Done

Late LDAP adapters quickly added inferior LDAP support to their products. Instead of having their products query LDAP directly for information, they decided to synchronize their products daily with an existing LDAP server and then pump the data into a database. Alas, their marketing literature could now exclaim “We support LDAP!”

Other vendors decided to use LDAP directly, but their schemas were ported from previous database-centric products and these directories could exclusively be used by them. This led to having multiple LDAP servers with the requirement for synchronization – one of the problems that LDAP was supposed to solve!

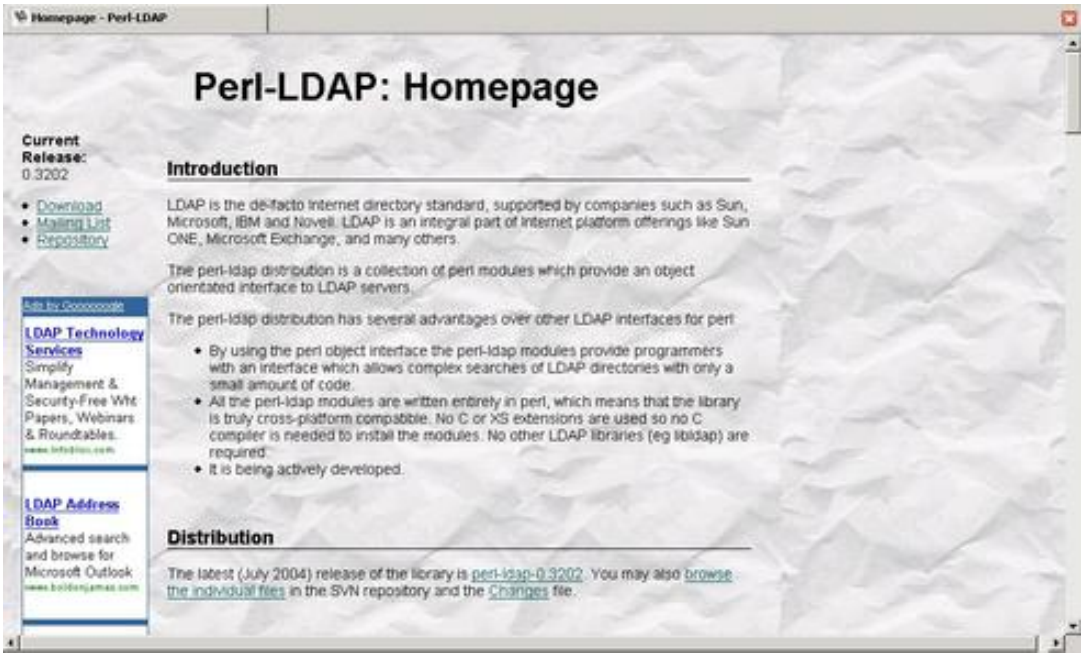
Today, some of these problems are overlooked (it is, after all, an open standard) and LDAP is a popular (and exposed) protocol. The problem now is that people who see the bad side of LDAP often fail to see how important it really is.

User interpretation

The freedom given to those who choose to deploy LDAP has ultimately led to problems in interoperability. DAP was largely ignored because too much time needed to be invested to plan an appropriate deployment strategy. Too much information would also have been needed to create a schema compliant user profile. LDAP left almost everything to the imagination. To quickly deploy LDAP and create user profiles, no planning was required. There were no standards for information (this would have been a hindrance initially and too close to X.500) and no best practices were provided. As with all new technologies, no one went far enough to have made any significant mistakes. Whoops.

A quick fix or temporary solution turned into an internal standard. When “Tom Jackiewicz” was created by different LDAP administrators in different environments, I could authenticate as *tom*, *tjackiewicz*, my badge number, *tjackiewicz* followed by my badge number, or *tjack* when the name was just too difficult to spell. While many of these were deployed in test environments, they were quickly adopted and became corporate standards that could not be easily changed to meet real needs within the environment. It was realized, that by choosing these naming standards, without any forethought, it would become difficult to integrate LDAP with other directories, databases, or data sources. Even the short-sighted deployment of a directory information tree (used to create branches within a flat directory) hurt integration efforts. At the top of the tree might be definitions for the whole internal user base. However, as LDAP is needed in other areas (such as external customers), or is used to store other data types, the lack of planning for the directory information tree causes problems when applying access controls or even setting appropriate search filters.

Fig. 3: Perl-LDAP provides a standard way of utilizing Perl to access your LDAP directory


Homepage - Perl-LDAP
Perl-LDAP: Homepage
Current Release:
0.3202
Download
Mailing List
Repository
Introduction
LDAP is the de-facto Internet directory standard, supported by companies such as Sun, Microsoft, IBM and Novell. LDAP is an integral part of Internet platform offerings like Sun ONE, Microsoft Exchange, and many others.
The perl-ldap distribution is a collection of perl modules which provide an object orientated interface to LDAP servers.
The perl-ldap distribution has several advantages over other LDAP interfaces for perl
By using the perl object interface the perl-ldap modules provide programmers with an interface which allows complex searches of LDAP directories with only a small amount of code.
All the perl-ldap modules are written entirely in perl, which means that the library is truly cross-platform compatible. No C or XS extensions are used so no C compiler is needed to install the modules. No other LDAP libraries (eg libldap) are required.
It is being actively developed.
Distribution
The latest (July 2004) release of the library is perl-ldap-0.3202. You may also browse the individual files in the SVN repository and the Changes file.
LDAP Technology Services
Simplify Management & Security-Free Wht Papers, Webinars & Roundtables.
LDAP Address Book
Advanced search and browse for Microsoft Outlook

Conclusion

The future of computing is currently in the hands of marketing departments and corporations. It has been pulled from the hands of the universities and computer scientists, innovating for the sake of doing what is right. What we must do, as a community, is insist on standards. Good yet proprietary ideas created by the vendors must be cherry picked and be turned into well-defined standards. We shouldn't let LDAP lose its simplicity and, ultimately, the reason we are using it in the first place. We should also make it clear to the vendors that we won't base our LDAP deployments on their systems. We want the ability to use whatever implementation we choose without having to conform to their ideas of what LDAP should be.

Bibliography

[1] Jackiewicz, Tom "Deploying OpenLDAP", Apress:2004

Copyright information

© 2005 by Tom Jackiewicz
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at http://www.gnu.org/copyleft/fdl.html

About the author

Tom Jackiewicz is currently responsible for global LDAP and email architecture at a Fortune 100 company. Over the past 12 years, he has worked on the email and LDAP capabilities of the Palm VII, helped architect many large scale ISP's servicing millions of active email users, and audited security for many Fortune 500 companies. Jackiewicz has held management, engineering, and consulting positions at Applied Materials, Motorola, and Winstar GoodNet. Jackiewicz has also published articles on network security and monitoring, IT infrastructure, Solaris, Linux, DNS, LDAP, and LDAP security. He lives in San Francisco's Mission neighborhood, where he relies on public transportation and a bicycle to transport himself to the office -fashionably late. He is the author of Deploying OpenLDAP, published by Apress in November 2004.

Running BSD on PowerPC/PPC

Using an alternative Unix on an alternative platform

Martin C Brown

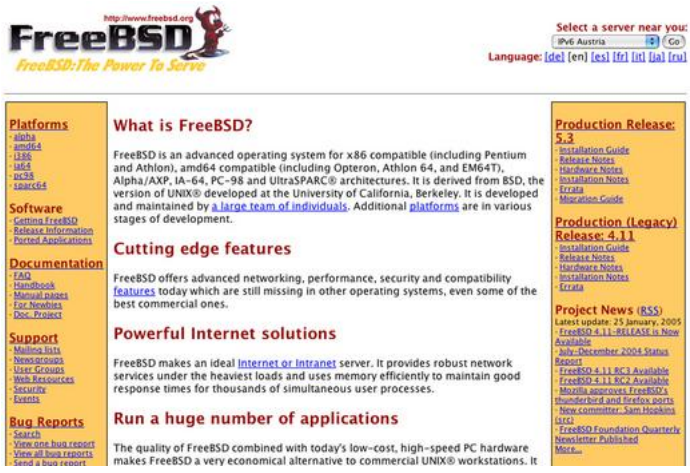
When you think of the PowerPC processor, chances are you'll think of just two platforms and, by association, two operating systems. Apple's Mac OS X, which runs on Apple's own hardware, and the AIX Unix operating system from IBM, which runs on their own PowerPC platform systems. In reality, there is a wide choice of potential operating systems that work on a wide range of PowerPC platforms. If you want a Unix-like alternative to AIX, particularly a free software one, then Linux seems the obvious choice, but there are others. BSD provides a Unix operating system, but with a heritage that goes back a lot further than Linux. If you chose to go down the BSD route, what solutions are available for the PowerPC platform? And, are they as good as Linux and proprietary alternatives?

The BSD operating system

All Unix flavors are typically based on the original version of Unix developed by AT&T. The BSD variant of Unix available now is a free software version of the AT&T Unix developed at the University of California, Berkeley. Proprietary and free software flavors of Unix available today, can all trace their heritage back to one of these two basic versions of the Unix operating system. Over the years different companies have based their Unix offering on either the AT&T (commonly referred to as AT&T System V Release 4 (SVR4)) or the BSD version.

There have also been a few changes over the years. For example, versions of Sun's SunOS operating system, up until SunOS 4.1.x, were based on the BSD code base, but then switched to an SVR4 base with SunOS 5.x, better known as Solaris. These changes are akin to a car manufacturer choosing a different engine manufacturer for the car – there are changes to certain parts of the internal workings, but the basics of internal combustion stay the same, and the outer shell to remains relatively unchanged.

Fig. 1: The FreeBSD website

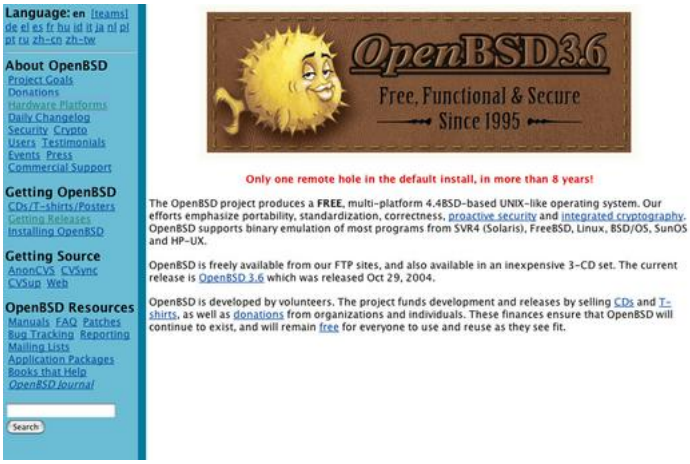

FreeBSD
http://www.freebsd.org
FreeBSD: The Power To Serve
Select a server near you:
IPv6 Austria
Go
Language: [de] [en] [es] [fr] [it] [ja] [ru]
Platforms
alpha
amd64
i386
ia64
pc98
sparc64
Software
Getting FreeBSD
Release Information
Ported Applications
Documentation
FAQ
Handbook
Manual pages
For Newbies
Doc. Project
Support
Mailing lists
Newsgroups
User Groups
Web Resources
Security
Events
Bug Reports
Search
View one bug report
View all bug reports
Send a bug report
What is FreeBSD?
FreeBSD is an advanced operating system for x86 compatible (including Pentium and Athlon), amd64 compatible (including Opteron, Athlon 64, and EM64T), Alpha/AXP, IA-64, PC-98 and UltraSPARC® architectures. It is derived from BSD, the version of UNIX® developed at the University of California, Berkeley. It is developed and maintained by a large team of individuals. Additional platforms are in various stages of development.
Cutting edge features
FreeBSD offers advanced networking, performance, security and compatibility features today which are still missing in other operating systems, even some of the best commercial ones.
Powerful Internet solutions
FreeBSD makes an ideal Internet or Intranet server. It provides robust network services under the heaviest loads and uses memory efficiently to maintain good response times for thousands of simultaneous user processes.
Run a huge number of applications
The quality of FreeBSD combined with today's low-cost, high-speed PC hardware makes FreeBSD a very economical alternative to commercial UNIX® workstations. It
Production Release: 5.3
Installation Guide
Release Notes
Hardware Notes
Installation Notes
Errata
Migration Guide
Production (Legacy) Release: 4.11
Installation Guide
Release Notes
Hardware Notes
Installation Notes
Errata
Project News (RSS)
Latest update: 25 January, 2005
FreeBSD 4.11-RELEASE is Now Available
July-December 2004 Status Report
FreeBSD 4.11 RC3 Available
FreeBSD 4.11 RC2 Available
Mozilla approves FreeBSD's thunderbird and firefox ports
New committer: Sam Hopkins (src)
FreeBSD Foundation Quarterly Newsletter Published
More...

Today, the most popular implementations of BSD are the free software ones, much the same as Linux. But BSD has that much older heritage and over the years certain aspects of the inner workings of BSD have been improved and expanded upon, particularly with respect to networking and security, generally making BSD more stable and giving it

a higher performance for certain operations. BSD has for many years had one of the best network stacks available often exceeding the nearest competitor on network throughput, utilization and connection initiations. Hackers and crackers are frequently challenged to break into BSD based systems and have no success.

This heritage in networking has meant that BSD is regularly used in routers and network servers and as a platform for web services. Unfortunately, it has failed to capture the same hype and following as Linux. Though, it does have its dedicated followers, including myself, for use as a general purpose Unix platform.

Fig. 2: The OpenBSD website


Language: en [teams]
de el es fr hu id it ja nl pl pt ru zh-cn zh-tw
About OpenBSD
Project Goals
Donations
Hardware Platforms
Daily Changelog
Security Crypto
Users Testimonials
Events Press
Commercial Support
Getting OpenBSD
CDs/T-shirts/Posters
Getting Releases
Installing OpenBSD
Getting Source
AnonCVS CVSync
CVSup Web
OpenBSD Resources
Manuals FAQ Patches
Bug Tracking Reporting
Mailing Lists
Application Packages
Books that Help
OpenBSD Journal
Search
OpenBSD3.6
Free, Functional & Secure
Since 1995
Only one remote hole in the default install, in more than 8 years!
The OpenBSD project produces a FREE, multi-platform 4.4BSD-based UNIX-like operating system. Our efforts emphasize portability, standardization, correctness, proactive security and integrated cryptography. OpenBSD supports binary emulation of most programs from SVR4 (Solaris), FreeBSD, Linux, BSD/OS, SunOS and HP-UX.
OpenBSD is freely available from our FTP sites, and also available in an inexpensive 3-CD set. The current release is OpenBSD 3.6 which was released Oct 29, 2004.
OpenBSD is developed by volunteers. The project funds development and releases by selling CDs and T-shirts, as well as donations from organizations and individuals. These finances ensure that OpenBSD will continue to exist, and will remain free for everyone to use and reuse as they see fit.

BSD variants

There are quite a few variants of the BSD operating system, all of which share a common foundation within the original Berkeley Systems Distribution edition of the operating system. The differences that exist are in the focus of the development, including performance, target platforms, flexibility and security.

There are four mainstream versions of BSD that most people are familiar with:

- FreeBSD (`http://www.freebsd.org`) - Focused on performance, FreeBSD is the most popular of the BSD solutions after being released by Walnut Creek CD-ROM, an early pioneer of freeware and shareware distributions before the internet became mainstream. FreeBSD has the largest user base, probably because of the size of the development team and the number of applications that have also been ported to the operating system. FreeBSD is not, at the time of writing, fully available on the PowerPC platform, although there is an active porting project in progress.
- NetBSD (`http://www.netbsd.org`) - Focused on portability, NetBSD is best known for it's wide-ranging hardware support. You can run NetBSD on everything from an embedded solution to proprietary hardware from IBM, Sun and others. If you need to support a Unix platform on different PowerPC based platforms including VME, evaluation boards, IBM pSeries and RS/6000 hardware, PowerPC based Macs and even old BeBoxes, then NetBSD is the solution for you. Beyond the hardware environment, NetBSD also has the same strong networking foundation of other BSD variants along with similar security and stability.
- OpenBSD (`http://www.openbsd.org`) - Focused on security, OpenBSD is technically a fork of the NetBSD code. Contrary to the name, OpenBSD is, upon initial installation, closed. All of the main network services are disabled until you enable them and much of the operating system is in a "switched off" state until the administrator says otherwise. This means that OpenBSD is, out of the box, incredibly secure; to use a network service, for example SSH, you'll need to specifically set up SSH, open the ports and configure the keys and background daemon, before using it to administer your system. Although this seems long-winded, it really does mean that as an administrator, you must explicitly allow potential security problems, rather than the implicit approach used on other operating systems. OpenBSD also includes extensive encryption and security options making it incredibly popular with network administrators for use in routers and web servers. OpenBSD is available for Apple's Mac-based PowerPC hardware, but not any other PowerPC platforms.
- Darwin (`http://developer.apple.com/darwin/`) - Darwin is produced by Apple and is available for free, although it's not heavily publicized. Darwin is based on FreeBSD (changes in Darwin are actually added back into the FreeBSD code base), but with additions and added functionality for use *on*, and

more importantly *with*, Apple systems. For example, Darwin incorporates functionality for integration with the Apple LDAP-based authentication system, and the necessary base libraries and components to enable you to use Darwin for QuickTime streaming. The server software is available on the Darwin homepage. The Darwin OS also forms the basis of Apple's proprietary Mac OS X operating system, when combined with the Cocoa user interface and other components, such as QuickTime. The benefit, from a user (and administrators) point of view, is that you get a rich GUI environment for applications like Microsoft Word and Adobe Photoshop, but with the underpinnings of a Mach kernel and BSD operating system that provide stability and security and make it easy to support applications like Apache and MySQL.

Aside from Darwin, the other three are licensed using Berkeley's free BSD License that allows complete access to, and use of, the code for any purpose. Darwin is licensed under the Apple Public Source License, which is a little more restrictive to help protect the work carried out by Apple on what is the underpinning of a proprietary product.

Comparing BSD to Linux

The key difference of BSD compared to Linux is the way the operating system is provided. Linux is technically nothing more than an operating system kernel. Without the other free utilities provided with it in a distribution (RedHat, SuSE etc), Linux would not be a usable operating system. This is the reason for the "distribution" in the first place; a distribution includes the Linux kernel and the other utilities packaged up into a unique bundle, often with an installer to make the process of getting the operating system working on your machine much easier.

Linux distributions are not a bad way of providing and supporting an operating system. They do mean, however, that you are relying on more than one source for much of the functionality of the core operating system and this can cause problems from simple compatibility right up to security holes and others. The sheer inter-related nature of Linux has also led to a number of different "package manager" solutions designed to simplify, or at least get you out of, some of the complexities associated with such a wide range of different software.

Fig. 3: The NetBSD website

On the other hand, BSD is a complete operating system with both the kernel and a whole range of utilities and components that make up the operating system. This makes BSD a more coherent operating system. BSD releases are made when all of the utilities and other components have been updated and patched. This fact also helps to explain why many of the BSD operating systems are more stable and secure than Linux. The coherence also means that BSD offerings are easier to use, manage and migrate between. Finally, because a BSD release is managed and controlled by a single group, rather than made up of numerous components from many contributors there's a lot of consistency between the operating system and it's documentation.

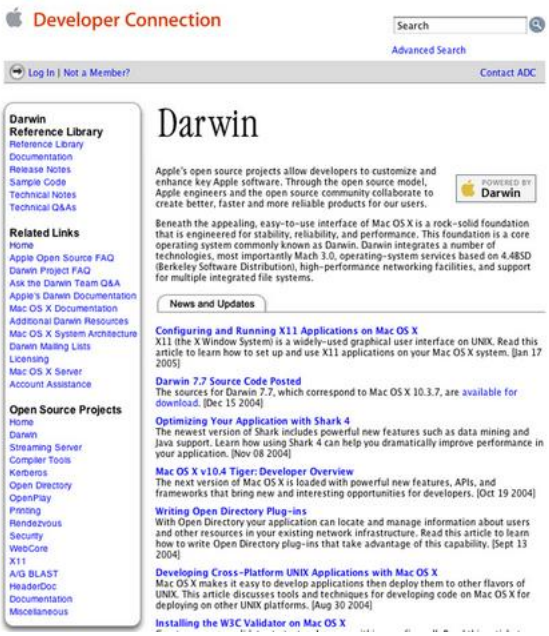
The BSD groups, although fanatical, also approach the problem of other operating systems by working with, rather than competing against them. With such a long history, BSD is generally more stable (many BSD administrations will quote uptimes in months, or even years, rather than days or weeks), and with better performance than Linux.

In terms of compatibility, most BSD distributions come with the facility to run Linux applications, which means that in many cases you can simply install and execute a Linux binary (providing it's for your target platform) without any further modification or configuration. Applications that are available in source form should configure and build fine. However, where Linux does have an advantage is that because it is the current favourite, many companies are providing native versions of proprietary applications for Linux. Compatibility mode in these situations is not always perfect. Having native BSD versions of these applications would be preferred.

Comparing BSD to proprietary offerings

The proprietary Unix sphere is much larger than most people think, but most will identify with the big names such as Solaris and AIX. Many would be hard pressed to actually name another proprietary Unix offering beyond these two. Most proprietary Unix variants are focused on a fixed platform; Solaris is primarily produced for Sun's own SPARC based hardware (although an Intel version is available), while AIX is specific to IBM's PowerPC platform.

Fig. 4: The Darwin project


Developer Connection
Search
Advanced Search
Log In | Not a Member?
Contact ADC
Darwin
Reference Library
Reference Library
Documentation
Release Notes
Sample Code
Technical Notes
Technical Q&As
Related Links
Home
Apple Open Source FAQ
Darwin Project FAQ
Ask the Darwin Team Q&A
Apple's Darwin Documentation
Mac OS X Documentation
Additional Darwin Resources
Mac OS X System Architecture
Darwin Mailing Lists
Licensing
Mac OS X Server
Account Assistance
Open Source Projects
Home
Darwin
Streaming Server
Compiler Tools
Kerberos
Open Directory
OpenPlay
Printing
Rendezvous
Security
WebCore
X11
A/G BLAST
HeaderDoc
Documentation
Miscellaneous
Darwin
Apple's open source projects allow developers to customize and enhance key Apple software. Through the open source model, Apple engineers and the open source community collaborate to create better, faster and more reliable products for our users.
POWERED BY Darwin
Beneath the appealing, easy-to-use interface of Mac OS X is a rock-solid foundation that is engineered for stability, reliability, and performance. This foundation is a core operating system commonly known as Darwin. Darwin integrates a number of technologies, most importantly Mach 3.0, operating-system services based on 4.4BSD (Berkeley Software Distribution), high-performance networking facilities, and support for multiple integrated file systems.
News and Updates
Configuring and Running X11 Applications on Mac OS X
X11 (the X Window System) is a widely-used graphical user interface on UNIX. Read this article to learn how to set up and use X11 applications on your Mac OS X system. [Jan 17 2005]
Darwin 7.7 Source Code Posted
The sources for Darwin 7.7, which correspond to Mac OS X 10.3.7, are available for download. [Dec 15 2004]
Optimizing Your Application with Shark 4
The newest version of Shark includes powerful new features such as data mining and Java support. Learn how using Shark 4 can help you dramatically improve performance in your application. [Nov 08 2004]
Mac OS X v10.4 Tiger: Developer Overview
The next version of Mac OS X is loaded with powerful new features, APIs, and frameworks that bring new and interesting opportunities for developers. [Oct 19 2004]
Writing Open Directory Plug-ins
With Open Directory your application can locate and manage information about users and other resources in your existing network infrastructure. Read this article to learn how to write Open Directory plug-ins that take advantage of this capability. [Sept 13 2004]
Developing Cross-Platform UNIX Applications with Mac OS X
Mac OS X makes it easy to develop applications then deploy them to other flavors of UNIX. This article discusses tools and techniques for developing code on Mac OS X for deploying on other UNIX platforms. [Aug 30 2004]
Installing the W3C Validator on Mac OS X

Actually comparing proprietary Unix variants is very difficult because many have used and exploited different components from both BSD and System V Release 4 (SVR4). Others have changed their spots half way through production (SunOS and Solaris, for example). The proprietary Unix variants also frequently provide either an SVR4 or BSD compatibility layer to help migration from one to the other, or to help users of other variants to migrate to the new platform.

Ultimately making comparisons boils down to examining the heritage and support of the operating systems and the platforms supported. There are two mainstream, proprietary Unix variants, which run on the PowerPC CPU, Mac OS X and AIX. As you know, Mac OS X is essentially Darwin, which is in turn based on the FreeBSD and only works on the apple PowerPC platform.

AIX is the Unix operating system produced by IBM and which runs only on IBM's own PowerPC based hardware. AIX is based on SVR4 and has some obvious advantages compared to both BSD and Mac OS X. The first is the availability of different software. As a proprietary Unix, it also has the best options available for proprietary support; IBM themselves provide many of their applications in an AIX version. Interestingly though, even IBM provide more of their software in Linux versions than AIX.

The other big advantage is the level of support offered by IBM for their AIX customers. You can get full, 24-hour support with an AIX expert if you are willing to pay for it. Support for most BSD offerings is provided on an informal email basis, which isn't suitable for customers using BSD in a proprietary environment. Choosing to use BSD, in a proprietary environment, will therefore depend on the experience of the people managing the system.

Installing and running BSD

Darwin is probably the easiest of the BSD options to install; neither of the OpenBSD and NetBSD offerings are difficult, but neither are straightforward for the average user. FreeBSD, when fully available on the PowerPC platform, should also be easy to use and install if the Intel experience is anything to go by.

Once installed, BSD operates much like any Unix variant; the familiar shells, command line utilities and other components are available to us. The big differences within the BSD system are the way in which you interact with and configure various components of the system. Everything - from the boot process through to the configuration of various components - differs from that in Linux and most proprietary distributions. There are also other differences in some of the command line options for different utilities; for example, the ps command (for listing processes) uses slightly different options to show all the running processes. Even so, the differences are minimal and while it takes some adjusting, if you are familiar with another Unix, the basic mechanics of interacting with the system are identical.

Even more significant though is the fact that the wide range of third-party applications and utilities that you are familiar with, Apache, the GNU tools, Emacs, Gimp and many others are all also compatible with the BSD environments.

Fig. 5: The FreeBSD website

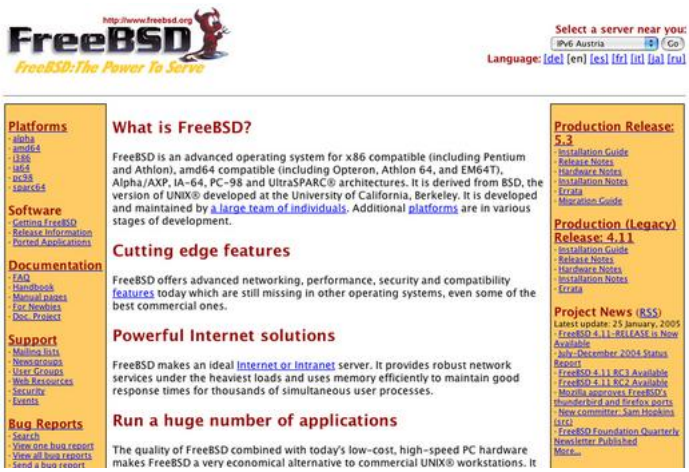

FreeBSD
FreeBSD:The Power To Serve
Select a server near you:
Language: [de] [en] [es] [fr] [it] [ja] [ru]
Platforms
Software
Documentation
Support
Bug Reports
What is FreeBSD?
FreeBSD is an advanced operating system for x86 compatible (including Pentium and Athlon), amd64 compatible (including Opteron, Athlon 64, and EM64T), Alpha/AXP, IA-64, PC-98 and UltraSPARC® architectures. It is derived from BSD, the version of UNIX® developed at the University of California, Berkeley. It is developed and maintained by a large team of individuals. Additional platforms are in various stages of development.
Cutting edge features
FreeBSD offers advanced networking, performance, security and compatibility features today which are still missing in other operating systems, even some of the best commercial ones.
Powerful Internet solutions
FreeBSD makes an ideal Internet or Intranet server. It provides robust network services under the heaviest loads and uses memory efficiently to maintain good response times for thousands of simultaneous user processes.
Run a huge number of applications
The quality of FreeBSD combined with today's low-cost, high-speed PC hardware makes FreeBSD a very economical alternative to commercial UNIX® workstations. It
Production Release: 5.3
Production (Legacy) Release: 4.11
Project News (RSS)

Fig. 6: An OS X Session, with Word and X Windows

XEmacs
Comparing BSD to Commercial
The commercial Unix sphere is much large
big names such as Solaris and AIX. Many
commercial Unix offering beyond these tw
platform; Solaris is primarily produced for
version is available), w
Actually comparing co
exploited different com
changed their spots hal
Unix variants also frequ
from one to the other,
Ultimately making com
systems and the platfor
on the PowerPC CPU,
is in turn based on the
AIX is the Unix operat
based hardware. AIX i
and Mac OS X. The fir
the best options availa
applications in an AIX
Linux versions than AI
The other big advantage is the level of support offered by IBM for their AIX customers. You can get full 24 hour support with an AIX expert if you are willing to pay for it. Support for most BSD offerings is provided on an informal email basis, which is not suitable for customers using BSD in a commercial environment. Choosing to use BSD use in a commercial environment will therefore

Choosing a BSD OS

If you were in the Intel space, choosing a BSD solution would be more straightforward. All of the BSD versions are available for the Intel space and choosing the solution that's right for you comes down to the type of installation and functionality that you require.

Instead, when limited to the PowerPC platform you need to base your decision on the hardware available to you. NetBSD has the major advantage, in that, it runs on just about everything, from embedded PowerPC platforms to the more mainstream solutions such as Apple Macs and IBM's PowerPC hardware platform such as the RS6000 desktops and servers.

OpenBSD is the best for security, but you are limited to Apple hardware. While this is not a real problem, since both new and old Apple PowerPC based kits are readily available, it is still fairly limiting. If Apple is your platform of choice though, Darwin offers a more rounded environment. Darwin also has the benefit of proprietary backing – which I think in the long term will aid its longevity and should also help publicize BSD at a time when Linux gets the column inches.

Conclusion

Despite what you might have been led to believe, Linux isn't the only alternative for a stable and secure Unix like operating system. BSD is just as accessible, easy to install and even compared to proprietary offerings has a much older heritage, and it's generally better because of it. The only limitation of BSD is the supported platforms. NetBSD is the best if you want the ultimate in compatibility across a range of platforms – including Intel and embedded solutions – but if your hardware is Apple based then OpenBSD is a security focused solution that is at home on the server or the desktop.

Copyright information

© 2005 by Martin C Brown

This article is made available under the "Attribution" Creative Commons License 2.0 available from http://creativecommons.org/licenses/by/2.0/.

About the author

Martin "MC" Brown is a freelance writer and consultant, he works with Microsoft as an SME, is the LAMP Technologies Editor for LinuxWorld magazine, a founding member of AnswerSquad.com, Technical Director of Foodware.net and, and has written books on topics as diverse as Microsoft Certification, iMacs, and free software programming. He can be contacted through his website at MCslp.com.

Guerrilla marketing part two

Getting good press coverage

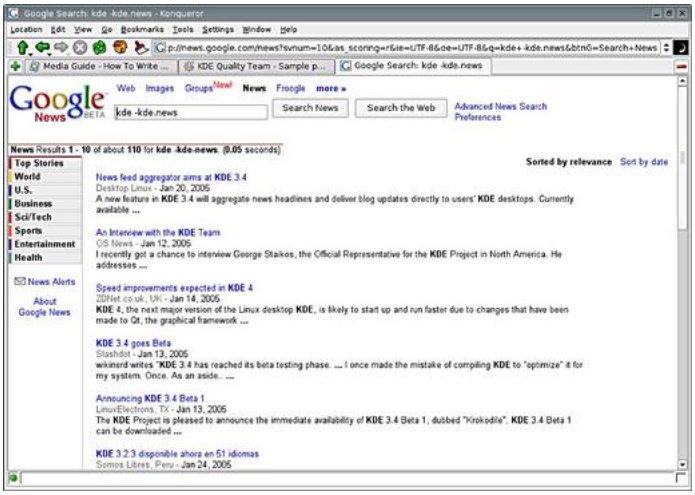
Tom Chance

Following on from my general introduction to guerilla marketing in the first issue of this magazine, I will now discuss some specifics of getting good press coverage. This much-neglected area of marketing is actually a relatively important issue, especially if your project is genuinely interesting, and can reap huge rewards.

Press coverage (a much-neglected area of marketing) is actually a relatively important issue, especially if your project is genuinely interesting and can reap huge rewards

Why would you want press coverage? This seemingly obvious question is worth considering before you embark into the media world. Do you want to raise awareness of your project to gain more users? Maybe you want to recruit more programmers, artists, documentation writers or other kinds of volunteers. You may want to improve the image of your project, or of the kind of work you're doing, for example, responding to criticisms or promoting free software in the mainstream media. You may also want to bring your project to the attention of commercial sponsors and software distributors. I may have my reservations about the media, this magazine excepted of course, but for the reasons above and more besides the media is a tool that you can use to your advantage.

Fig. 1: A good media strategy will get you regular coverage



Planning an effective strategy

Assuming you have followed the marketing strategy checklist I described in the first part of this series, you should have a good idea of what you want to say about your product and who you are targeting. You may also have assigned press roles to volunteers in your project. You now need to convert this very general strategy into one specific for the media, which will involve the following considerations:

Coming up with an effective strategy isn't an exact science. In working through the above questions, you should come to better understand how you are going to approach the practical tasks of media work, including writing press releases, following them up, taking part in or conducting interviews

Textbox 1: Media strategy considerations

Why do you want media coverage?
Who are the target audience?
What type of coverage does the project need to receive?
Which media are you targeting?
How do you get them interested?
What are my key messages?

and even writing your own articles. The process is perhaps best illustrated with an example.

You should come to better understand how you are going to approach the practical tasks of media work, including writing press releases, following them up, taking part in or conducting interviews and even writing your own articles

I am the press officer for the Gneb project, a GNOME web development application. I want press coverage because the project could do with a few more volunteers, and because I want to raise awareness of our upcoming release. I'm therefore trying to reach all of our existing and potential users - so anyone interested in web development - and in particular those with the skills and inclination to volunteer. I want to receive lots of media coverage that emphasises the benefits and advantages of our application over similar free software web development applications, and that suggests people should become involved in the project. To achieve all of this, I will target the usual free software-friendly media, and trade magazines that discuss web development. I'll get them interested by emphasising the great features of our application that set it apart from the rest, and by showing the mainstream press the value of free software. My key messages are that the application makes managing complex web frameworks easier, that it integrates perfectly with the GNOME desktop, and that we're keen to attract new volunteers.

Obviously the above example is a little contrived and light on detail, but you get the idea; if you don't do this, you're far less likely to attract the attention of journalists, and even less likely to get really good coverage. Preparation can mean the difference between no coverage, a small mention and a full feature article. Ask yourself why a journalist would really want to write about *yet another* web development application. Perhaps the best way to learn how to use the media is to read, listen to and watch the media more critically. Next time you read a feature article or an interesting piece of news, reflect on why the journalist chose to cover it.

Announcements and press releases

Announcements and press releases are the most basic way to promote an application. Announcements can grab the attention of users, showing them why a new release is worth installing, or why they should get involved in your project, or they can show them what your aims are. They can be reasonably long, and discuss all the messages you want to promote in detail. A really good announcement will tear me away from my game of Freeciv and make me spend a few minutes reading through it.

Press releases catch the attention of editors and journalists, they make them research and write an article or do a piece in a show about your application. Editors and journalists can receive hundreds or even thousands a day, so they'll most likely look over a press release extremely quickly, and discard it if it doesn't interest them. You can either use a press release as the basis for an article, giving the journalist an easy job and ensuring positive coverage, or you can use it as a way of getting them to contact you for a fuller story. You can also use them to point towards the (more verbose) announcement, allowing your press release to be short whilst giving them lots of information.

Let's start with a brief look at the process involved, before going into each stage. You should write the announcement first, then the press release that accompanies it; next, you send the press release to all relevant contacts, following this up a few days later where appropriate; finally, you record any useful contacts and make a note of interesting leads.

Writing the announcement and press release

Announcements are fairly easy to write. You should start with a paragraph of no more than a few sentences summarising your key messages. You then flesh out each of

Fig. 2: Follow this process for announcements press releases

Write your announcement & press release, then post it to your web site
Email and fax the press releaase to editors and friendly journalists
Press
Follow it up with a phonecall to contacts likely to cover your announcement
Record any useful contacts you have gained

these messages, for example by providing a bullet point list of your most important and interesting new features, or by describing the kinds of tasks volunteers might get involved with. You can optionally finish it off with some boasts, telling the reader where your application is being used or how much more efficient it is than your competitor's. Don't sound *too* smug, otherwise you'll put people off, but *do* give the reader the impression that there is something special or different about your project and product.

Press releases are much harder. Whereas an announcement just needs to be reasonably well written and cover the key messages, according to your media strategy, the press release needs to do it with *precision* and *concision*, so that at a brief glance the key messages grab your attention. Here's an example press release that I'll work through:

The press release needs to start with a snappy title. Now I'll admit I'm not great at this, so don't worry if you can't think of anything clever; so long as it is reasonably interesting it should avoid being deleted from the journalist's inbox immediately. The second line should state the release date or, if appropriate, an embargo that tells the journalists not to use the information in the release until the date specified. Embargoes aren't binding, and may irritate the journalist or put them off, so don't use this unless you have a good reason.

The first paragraphs are as important as the title. All your key messages should be here to hook the reader, summarising the rest of the story. You can then use your announcement to concisely expand upon these messages in the next

Fig. 3: An example press release

The Gneb Project
http://www.gneb-project.org

New Gneb release gives great web performance

PRESS RELEASE

FOR IMMEDIATE RELEASE / EMBARGO: 27th January 2005

The Gneb project is to announce the release of Gneb version 1.0, the indispensable tool for all web developers, featuring advanced project managament and full integration with GNOME. The Gneb project is seeking new volunteers to help develop on this stable platform in the future.

Gneb is the result of two years of development by a team of over twenty volunteers. This release makes the application complete, putting ahead of the competition in terms of features, stability and integration with the desktop. Based upon the free software platforms GNU/Linux, GNOME and Gecko, Gneb provides powerful tools for handling multiple library and include files, synchronising the work of a team of developers and writing cheese with ease. Gneb also provides syntax highlighting, code folding and other features common to modern web development applications.

The Gneb project is run by volunteers, and so the more participation we receive the better the product can become. We are keen to recruit new volunteers, in particular from companies, to work on the code and documentation. Those interested should contact Tom Chance [2].

Tom Chance, Gned's Project Coordinator, commented that:

"Gneb is now being used in hundreds of companies around the world, and with our competitive pricing model for custom feature requests, you can get your ideal platform at a fraction of the cost of competing products"

ENDS

Notes for editors

1. For the full announcement of this release, and more information on Gneb, visit: http://www.gneb-project.org

2. For more information, contact Tom Chance:
Email: tom@fakedomain.com
Phone: +44 (0)1234 567890

few paragraphs. It's important that these don't give the journalist too much or too little information, and that it is accurate. Leave grand or vague claims for the quote, which comes next. Stories with quotes look much more interesting than prose from the journalist, so make up a nice soundbite that the journalist can't misrepresent.

Finally, you should provide any further information in notes. These can include links to the full announcement, links or more information on any points you made concisely in the text but that can be expanded upon, background information (e.g. about your project). Most importantly, you should include contact details; if you put in a phone number, make sure you are always available on it, because a journalist is unlikely to chase your answer phone.

Making contact and following up

By now you should have a reasonable announcement and press release written and posted on your web site. Now you need to get the media to pay attention. Start by putting together a list of people you can send the release to. Look for editors' addresses in magazines and on web sites; find press contacts; hunt around for journalists that have covered your application or ones like it in the past; find all the web sites with news submission forms; phone the switchboard

for newspapers, radio & TV stations and any other media outlets and ask for the newsdesk.

After some groundwork, you should have a decent contact list to start with. It's also worth looking into deadlines, to make sure you get the news to them in time. If a weekly paper goes to print on Friday you'll want to get the release to them by Wednesday midday at the latest so they have time to write the article.

> Preparation can mean the difference between no coverage, a small mention and a full feature article. Ask yourself why a journalist would really want to write about *yet another* web development application

Now email the press release out. Either email it to each contact individually (scripting comes in handy here) or send it to yourself and put their addresses in the BCC (Blind Carbon Copy) field to avoid annoying them all. If you are able, send it out by fax too on nice headed paper as in my example release above. When sending to individual journalists, you are more likely to win their attention with a personal email than with one that has obviously been bulk-mailed out to a contact list.

If all goes well you should now have journalists browsing your web site to find out more, or even reaching for their phone to call you. But to make sure you can follow the press release up a short while later to check that they have received it and to see if they have any questions. Calling media folk in this way may make you feel like a door to door toothbrush salesman, but there is no alternative. Media organisations are chaotic, and things get lost. When the newsdesk says "never seen it before", they may even be telling the truth. So send it again.

Lather, rinse, repeat

Congratulations, by now you have made your first brave steps into the media jungle. You've got coverage in all of your target publications, and users and volunteers are flooding in. But the work's not over yet. You're likely to have other announcements in the future, so spend a little time gathering together contacts and materials from this round. Record the contact details of all the journalists that did cover you in your contact book, marking those whose articles you liked as "sympathetic reporters", with a reference to where the article they wrote is. Keep clippings too, both for your personal use and for promotion (e.g. to show off on stalls at trade shows).

> With a growing contact book, a well organised archive of announcements and press releases, a readily available press contact and an interesting project, you can't go wrong

With a growing contact book, a well organised archive of announcements and press releases, a readily available press contact and an interesting project, you can't go wrong. You're not guaranteed to hit the front page of national newspapers, but as you learn to target quiet spots in the media cycle (e.g. public holidays, late summer) and as your press releases become more journalist-friendly, your profile will grow and grow.

Copyright information

© 2004 by Tom Chance

This article is made available under the "Attribution-NonCommercial" Creative Commons License 2.0 available from http://creativecommons.org/licenses/by-nc/2.0/.

About the author

Tom Chance is a philosophy student, free software advocate and writer. He has worked on promotion with the KDE Project, acting as press officer at their annual conference, aKademy, in 2004. You can contact him via his web site (`http://www.tomchance.org.uk`)

Open Source Labs

Pioneers in Open Source Solutions

"Open Source Labs, a division of VYOM Labs, is a centre of excellence providing World-Class Support, Consultancy and Training across the entire spectrum of Open Source Technologies. Open Source Labs enables organisations worldwide to gain maximum value from Open Source Software while reducing the total cost of ownership".

Open Source Support Services

We provide migration, installation and post installation support solutions for most Open Source Technologies :

- Linux and MySQL
- Software Engineering tools like CVS, BugZilla
- Software IT management tools like Nagios, MRTG, OTRS
- Messaging solutions including Qmail, Sendmail, Webmail, SpamAssasin, Thunderbird
- Custom Open Source development in PHP, Perl (using MySQL or Postgres SQL servers)

Linux Training

- Linux OS Administration
- Networking and Security
- RedHat Linux
- SuSe Linux

Developer Course on Linux

- OSL 101 Unix Shell scripting
- OSL 102 PHP/Perl + MySQL

Open Source Labs
power of freedom

redhat®
TRAINING PARTNER
Authorized Red Hat Training Partner

vyomlabs
Looking beyond the Horizon

Open Source Labs, (A division of VYOM labs Pvt. Ltd.), Dayaprabha House, Opp Sulzer India, ITI Road, Aundh, Pune 411007, INDIA. Ph: +91 982278700, +91 20 2588 9236
e- mail : contactus@opensource-labs.com

www.opensource-labs.com

Paradigm Plus

Open source is a market force

How open source has changed and continues to change the IT market, and the behaviours of those who target it

Malcolm D. Spence

Companies seek to counter their competition in a variety of ways - pricing, packaging, branding, etc. There are a lot of options and any good product manager will know them well. One of the toughest situations to be in is that of competition, with the usual responses, when you are up against a competitor using the natural forces of the market place. Marketers refer to these tectonic shifts as "shocks", because they are unusual events. They change the way the game is played, and once that change is made, there is usually no going back. The change is structural. Discerning whether the change is strategic and a market force, or just a tactical move by competitor can be difficult at first. The majority of the time change is gradual. It often takes hindsight to reveal it as a fundamental shift.

Open source is not a tactical move by some product company, to obtain some temporary advantage. It is a basic change in the way the users, most often software engineers, want to assemble their systems. It satisfies a hunger for changes to the way proprietary software has been used in projects.

The majority of the time change is gradual. It often takes hindsight to reveal it as a fundamental shift

Nor is open source a religious experience. Practical hybrid business models, which combine both open and proprietary elements, have already emerged and both coexist. Indeed they can compliment each other for quite a while. Competition will improve both models, and consequently improve those that employ a mix.

If the truth is to be told, the proprietary software companies have not quite figured out how to respond to the open source challenge to their business model. They are used to pitting product feature against product feature, or matching price with price. The process and the business models of open source are both completely different from their previous experiences. It's not as easy to respond, in kind, with that type of situation. Plus market forces are rapidly shaping the open source market such that it is now becoming an accepted part of the landscape. It is no longer strange and different.

The purpose of this article is to explore how we have arrived at this point, and what steps along the way should have told us that this is where we would be, at this point in time. Large software companies that are proponents of only proprietary software are fighting against an extremely powerful opponent, the natural force of a market in transition.

Some background on market forces

We are a long way from 1776 when Adam Smith first published his groundbreaking book "The Wealth of Nations". It became the basis of modern political economic understanding. Software and information technology (IT) may not have been a consideration when it was written but the basic rules still apply. The characteristics of software and

IT are also enabling their own change, which reinforces the notion of the market being adaptive.

Quoting from Adam Smith's book:

> Every individual necessarily labors to render the annual revenue of the society as great as he can. He generally, indeed, neither intends to promote the public interest, nor knows how much he is promoting it. He intends only his own gain, and he is in this, as in many other cases, led by an *invisible hand* to promote an end which was no part of his intention. By pursuing his own interest he frequently promotes that of the society more effectually than when he really intends to promote it. I have never known much good done by those who affected to trade for the public good.

Adam Smith captured the essence of the market (and open source), i.e. self-interest by the consumer. This self-interest is the engine of the economy and competition its governor. He also mentioned the "invisible hand". Just like the wind, we can see its effects but cannot quite view it at work.

Adam Smith captured the essence of the market (and open source), i.e. self-interest by the consumer. This self-interest is the engine of the economy and competition its governor

Fundamentally:

1. Adam Smith explained how prices are kept from ranging arbitrarily away from the actual cost of producing a good.
2. Adam Smith explained how society can induce its producers of commodities to provide it with what it wants.
3. Adam Smith pointed out why it is that high prices are in effect a self-curing disease, because they encourage production of the commodity to increase.
4. Adam Smith accounted for a basic similarity of incomes at each level of the great producing strata of the nation. He described, in the mechanism of the market, a self-regulating system, which provides for society's orderly provision of goods and services. (*Comment:* This now applies in a global sense, and could be the basis of a similar article stating that the off-shoring advantages will not last long.)

This article will focus this discussion on points 1 and 2 and how technology itself has caused open source to be a viable option for satisfying demand and getting the market in balance.

Price versus cost

Software is a peculiar product. Unlike physical goods (like cars and TVs), it requires very little in the way of consumable (physical) resources to produce. At one point it was taken from magnetic media, copied to another media (floppies, or tape) combined in a box with printed documentation (derived, itself, from electronic files), shrink-wrapped and shipped via trucks from warehouses. Not exactly an optimum distribution method for something that starts off as, and then ends up in, an electronic format. A format that should be able to distribute the product at the speed of light, and where the distance of delivery is not a cost factor.

Software should be an *optimum good* in a connected world

Software should be an *optimum good* in a connected world.

In electronic format, software:

- Costs very little to store.
- Costs very little to deliver.
- Can easily meet a surge in demand.
- Offers almost zero latency between the act of selection and product reception.
- Offers replication at very little incremental cost and with no depletion of itself.

It fits well in a world of instant gratification. You see it, you want it… well you just got it!

Software itself costs labor to create and maintain. And then of course there is the cost of physical resources. There are computers, networking infrastructure, disk drives, and a building perhaps, for the developers to use. These are initial costs and once incurred the ongoing cost is often less.

Other significant costs, for traditional software products, are those of sales and marketing. Look at any high tech company and those are indeed significant, often being 40 to 60 per cent of revenues. Proprietary licensing models have to recover that cost quickly. The "cost of sales" is now an area where the costs incurred are seen as being of dubious value to the sophisticated client. Many companies now try to sell and deliver products over the internet as a way to use more of a "pull" model (with users buying), rather than the "push" model (of salespeople selling), to thus reduce their costs.

The point I'm trying to make here, is that there is often a big difference between software pricing and product cost. According to Adam Smith, that difference causes tension between buyer and seller. It also provides room for a competitor to insert their product at a new price point. As mentioned in the previous paragraph, the internet has enabled low cost ordering and delivery, by what is called disintermediation. That is, the elimination of the middle-men. This strategy drastically reduced selling costs, as long as your clientele were users of the internet and felt comfortable about buying products and services using it. With IT users that was, indeed, already the case.

It was apparent to the invisible hand of the market that the only new price point that could be employed successfully was that of zero. Anything else could be countered by short-term price cuts

When it came to software, Microsoft had entered the market at a lower price point than their predecessors such as the mini-computer and workstation manufacturers. This was predicated on the model of a PC being both low-powered and for a single user from a licensing perspective. The lower Microsoft price points, were more than compensated for by volume. (In fact for a while their lower prices made them heroes to IT managers who could use them to negotiate with their other vendors.) Products such as Microsoft's SQL Server held the rest of the market (Oracle's DB for example) in check. Microsoft's product costs (packaging and distribution) were, and still are, minimal compared to their prices. In the past, if a competitor tried to attack them through lower cost pricing Microsoft could always heavily discount its price, give away parts of its product functionality, or use its access to a broad product range to bundle a product (which might be under market share pressure), in such a way that: a competitor making a discrete or narrower range of products could not respond in kind. Case in point: Netscape and the browser wars, or today, the spam filtering and virus blocking software being added to MS Windows, at "no additional cost".

Initially the Microsoft proportion of the overall systems cost was low. As hardware prices retreated, under the assault of competition, the Microsoft prices were maintained, and their proportion of the overall cost increased.

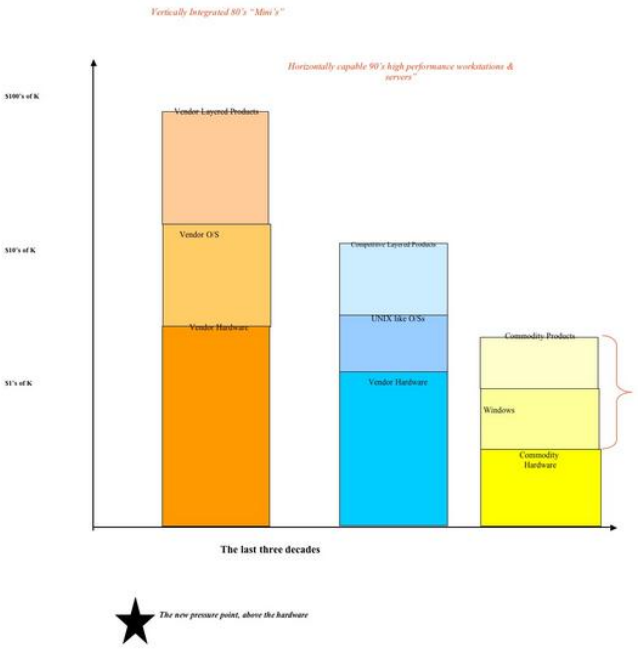
It was apparent to the invisible hand of the market that the only new price point that could be employed successfully was that of zero. Anything else could be countered by short-term price cuts. A case of "how low can you go?" And Microsoft and Oracle have deeper pockets. The only way Microsoft and other large software vendors might be vulnerable was by "death from a thousand cuts". That meant lots of small vendors nibbling away at their product lines by attacking across all fronts. A difficult proposition, as long as Microsoft had the operating system market to itself. It could always retreat its products into the safety of its OS. Plus switching costs, those that are incurred as users moved to new products, had to be offset by an overwhelming new value proposition.

Structurally the Microsoft business model is more dependent on license revenue. It has not yet switched to a service model. (Unlike say, IBM and its Global Services.) Hence the savageness of their counter-attacks to the whole notion of zero-cost licensing and open source. The key location now, on the value chain, for many companies and their software products is further downstream. After the delivery, it is the service that counts, as my local automobile dealership likes to say in its ads. (And my dealer now sells more than one brand of car!)

The solution stack

Customers don't just buy software. They often need a "stack" to make it all work. In general they need hardware, an operating system, a communication mechanism, perhaps a storage solution (i.e. DBMS), and finally a purchased or in–house developed application suite. Stack control is something vendors, (both hardware and software), have

Fig. 1: How the price of the software stack has evolved over the last three decades



wrestled over, with advantage shifting in the last decade to the operating systems vendors.

Stack control is something vendors, (both hardware and software), have wrestled over, with advantage shifting in the last decade to the operating systems vendors

Readers of this article may be too young to know, but in the 80's the mini-computer manufacturers (DEC, DG, Prime, etc) created a brand new market and competed with the much higher priced IBM mainframes. Their value proposition was a lower "cost of entry" for those companies that needed to use computers. They targeted the enterprise needs of medium sized businesses and the departmental level computing needs of the Fortune 1000 companies. They replicated the model of the mainframes and tried to provide vertically integrated solutions. (A solution in those days was likely to be FORTRAN on a VAX!) All users were trapped on the underlying hardware. This tight coupling enabled the vendors to move their margins around, using bundled prices. Compilers, development tools, even the operating systems were all priced attractively to enable clients to keep their high-priced hardware investment protected. Hardware services and upgrades were high-margin, cash cows.

Third-party software vendors often rounded out the solution stack, but even they were trapped. They had to minimize the range of supported platforms for their products, *as their developer skills were not sufficiently portable*. Understanding how important skill portability was, Oracle built a business model, trying to ensure that at least its users were masked from many of the variables of proprietary platforms, by creating a database product that was comparable from a developer viewpoint and yet spanned many platforms. But it was hard work. However, there was value for their clients and Oracle priced their product accordingly. By having Oracle, as part of the solution set, companies could demand better prices from the hardware vendors and more than offset the Oracle pricing premium.

The users, their third-party hardware vendors (disks, memory, printers, terminals etc.), and their third-party software vendors, all knew they were unable to grow and leverage competition, with the major system vendors locking in their customer base. Through pressure, and the role of IT standards, for both hardware and software, key pieces of the monolithic systems began to fracture. Vendors were forced to support both their own standards and public standards. Eventually public standards dominated selection criteria. Users forced vendors to remove margins from their hardware pricing. Luckily, manufacturing technology and VLSI had combined to create margins that could be reduced significantly. Lower pricing led to market growth. Additionally, the larger market, with more standards, and some commodity priced components, reduced the risk and entry cost for new competitors. With bigger pie, lower prices, more choice, the market was hard at work. Those who attempted to protect their market, shrunk. Those who exploited the new standards grew, albeit with reduced margins. Value was being created for the consumer.

This article is about open source software, so let's focus on some of the software aspects.

In the 90's UNIX had transformed many third-party software and in house development teams. Thanks to Bell Labs there was now a multi-hardware, somewhat open, operating system. As implemented by the vendors it was still not quite a uniform operating system, but sufficient enough to encourage its use as a primary target. The consequence of

this was that, operating system related skills now became more portable. Switching costs (amongst competing vendor UNIX versions) were now reduced for users. Switching software assets to new platforms was a reduced porting activity, as opposed to a major code rewrite. Third-party software developers could support more platforms for the same amount of effort. Vendors still tried to differentiate their O/Ss but they no longer had a lock. Price points were dropping for all products.

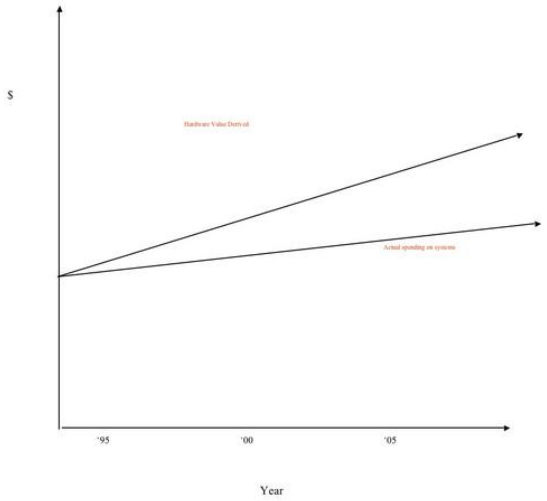
At this time a new phenomenon in software was also occurring. Universities and government research agencies were offering their code to other researchers and users for a nominal fee, usually the cost of the media and shipping. Such software usually ran on UNIX, (BSD UNIX was a preferred version), was unsupported, was licensed as "public domain" and expected a level of sophistication by the user if it was to be leveraged. Nevertheless the principle of sharing knowledge by sharing code had taken hold. This behaviour was reinforced by the hundreds of years of academic tradition in the sharing of and building on each other's knowledge.

In the late 80's IBM opened up the hardware architecture of PC and perhaps unwittingly not only created a new market, i.e. the desktop, but also offered hardware suppliers a level playing field in a growing market segment. The volume of PC component parts would now move hardware to its commodity phase. The third phase of the declining cost of systems was underway, and was now hardware driven. The OS was still a lock as far as Microsoft was concerned so they could leverage their position of power. They marketed their position in the middle as an island of stability, keeping hardware and software vendors in line with their APIs and compliance requirements. However in the 90's as both hardware and layered software prices continued downwards, under competitive pressures, the Microsoft offerings became more visible as a larger percentage of the solution stack. Their position in the stack made them now a target. Even hardware vendors, who had grown from nowhere in the previous 10 years, resented that most of the revenue dollars for systems shipped now went to Microsoft, who seemed immune to pricing pressures.

IT was still creating value, year after year. The US *Bureau of Economic Statistics* still showed tremendous value was being derived from the IT dollar, as shown below.

Their numbers were still demonstrating that since 1996 a 40% increase in spending had resulted in 96% improvements in systems capability. Nearly all the value was coming from manufacturing efficiencies in systems hardware including networking, products and services. (Networking value was coming about as the telecom industry was deregulated. Another seismic market shock!)

Fig. 2: The value derived from the IT dollar



What would be the next big driver of value?

Well for a start, the systems vendors now had some options. Windows NT was supposed to be the UNIX killer. But it did nothing of the kind. What's more UNIX had transitioned and extended from vendor chips (IBM, SGI, Sun, Digital) to running on hardware that previously had been Microsoft's exclusive domain, (known as Wintel). Sun's "Solaris on Intel" had made a brief run at the Microsoft position on the Intel architecture, but suspicions about Sun's long term commitment (as later justified by the brief time into its life when it appeared that it might be discontinued) and pricing model made it stall. SCO's UNIX ran on the Intel architecture, but its pricing model seemed intended to compete with the likes on AIX on PPC, Solaris on SPARC, and Tru64 on Alpha. It was not a direct threat to the Microsoft price point. BSD UNIX had been around for a long time but seemed unable to get real market traction. It seemed as though something else was needed to give it that nudge over the top. Linux was just that. It was gathering momentum at the right time. It was new and fresh, without the AT & T history and intellectual property encumbrances, was open source, and it also

had a lead developer in Linus Torvalds. He removed some of the anonymity of UNIX and gave it a face and a voice. He was somewhat unassuming compared with the boisterous and gregarious spokespeople of Microsoft, Oracle, Sun, HP etc. Software developers could identify with him.

Linux, combined with other open source layered products, was also creating its own solution stack. This was not a consumer oriented solution stack but was initially targeted at the software engineers assembling technical solutions and tailoring the stack. All this was facilitated by the internet and the low cost hardware resulting from the Wintel solution stack, which was forever demanding more memory, faster chips, and more storage. The irony was that systems that could no longer run the Wintel combo were finding new life with Linux. Linux and its apps were not as resource intensive. Linux could better protect a user's hardware investments. In order to use Linux you did not need new hardware, you could deploy on systems that were being retired from Windows use. Linux was freeing up capital for other uses closer to the users needs.

It's difficult for the vendors to attack open source products. You don't win friends and clients by telling them they have an ugly child!

In the meantime the ease with which open source projects could grow was now a force. The market was organizing an assault on the software stack. Below is a comparison of the old model of open source with today's open source development model.

Old

- Share ware/freeware/Public Domain.
- Cost of Media – 1800 BPI tape.
- Unsupported - use at your risk.
- Government research – most likely 1st generation code.
- Government Projects (NASA) – useful only for narrow markets.
- University orientation – knowledge flowed freely (as long as you were referenced).
- Geographically tight teams – not yet a global village.
- Lacking in infrastructure - expensive resources and vendor donated equipment came with strings attached.
- Went stale quickly, once research funds ran out.
- Pretty much shunned by business.

New

- Business strategy by major players.
- Carefully licensed to ensure the code base grows.
- Professionally packaged / supported / frequent release cycles.
- Service oriented - value and cost shift back to user control.
- Sophisticated development teams who are also users.
- Infrastructure enabled.

This last point meant:

1. Good & plentiful standards / lessons learned.
2. Email, instant messaging, free machine cycles, self documenting tools.
3. Newsgroups, SourceForge, Slashdot, virtual communities, access to experts.
4. Fast reliable download sites, mirrors, easy network access, plenty of bandwidth.
5. O'Reilly publishing.

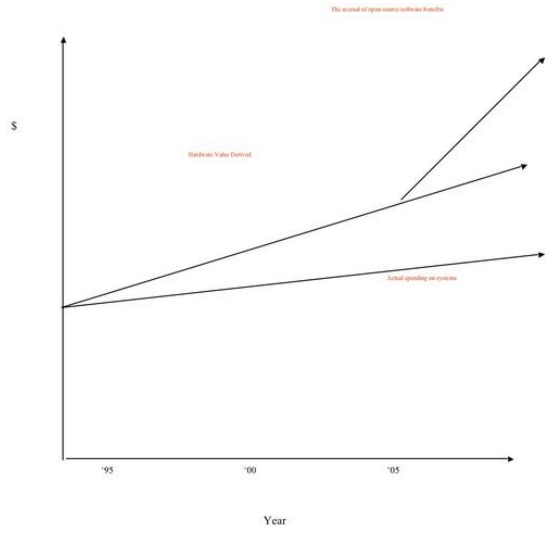
Scratching that Itch

Developers had always had itches. Now they had the collective means to scratch.

In regards to fighting back, the vendors were in a conundrum. The large proprietary software vendors were used to attacking their competitors (usually other vendors) with fear, uncertainty and doubt. When users, who were their customers, became collectively, their competitors, then things became a lot different. It's difficult for the vendors to attack open source products. You don't win friends and clients by telling them they have an ugly child! If we are a society that values ownership, then surely we should feel more ownership of our open source projects than in something we licensed. Licensing doesn't convey a feeling of ownership. Quite the opposite, it is restrictive.

Furthermore, it wasn't easy for proprietary vendors to compete once an open source product was in use at a company. Proprietary vendors need a very compelling value

Fig. 3: What the new productivity curve for information technology systems will look like (in my opinion)



proposition for their products, in order to convince someone that something is so much better that they should now pay money for it. In essence, the competition point now shifted down the value chain to service and the cost of use. In that regards, Microsoft attacking Linux by focusing on total cost ownership (TCO) is the right approach. That's the issue now. However, with Microsoft employing the wrong math it can boomerang. Conversely the open source community has to ensure it does not relax on its zero license cost laurels. Their products might be free to acquire, but have usage costs and have to earn their keep every day.

Companies deploying open source solutions will derive improved yields from their IT dollars. The major new value from information systems will come from leveraging software, because of its position in the stack. Hardware has very little left to give. Open source has lot to offer because of its low entry cost and zero cost of replication. Therefore the implementers who have the most to gain are those with large scale systems, i.e. those projects that would usually require large numbers of licenses and thus fees. Previously large scale users paid a penalty as they replicated the same solution across many systems. Now conversely, they can benefit the greatest by deploying widely, but paying only for the services they need, one time. They are scaling up the value; but scaling down the cost.

Ironically, once the decision to switch to an open source product is made, it behoves management to leverage it for as much value as they can. The community is now supporting it. Every home-grown application, or feature, that isn't strategic to their business, that they can shift into the community, results in lower internal costs.

The new software market model

In summary, the market is moving more into equilibrium, between buyer and seller. In the past, the software vendor held the advantage. Only the vendors knew and understood their products. They delivered binaries and patented or copyrighted their product to preclude you from working with its internals.

Open source shifts the pricing power back to the users and shifts the value downstream to services

Open source shifts the pricing power back to the users and shifts the value downstream to services. The new market will react to the following drivers:

- Good software engineering knowledge and practices are becoming more widespread. People are no longer intimidated by technology.
- More standards, which will minimize overall risk. Most importantly though, they will emerge earlier in the technology cycle, where they can help more.
- More prevalent, well-documented, software patterns solving re-occurring problems. They will be modeled, coded and ready for use in public community groups. Public components will be the new piece-parts of software systems.
- OO software continues to dominate with its support for easy extensions to applications, because of its ability to mask what you do not need to know, and enable you to satisfy your unique needs.

Open source projects will attract more companies offering commercial support. Their pricing power will be held in check by: 1) Competition amongst the support community itself (newsgroups); 2) The ability of end users to support themselves, should they elect to do so; and 3) The emergence of tiered, open source support, priced above the free

model of the newsgroup, i.e. low cost regional providers, and optional higher cost, higher quality, global providers.

Some observations on the last three items:

- Free support from the various communities is nice. It can get you going, but it's not timely or predictable, and it's often only one helpful person's opinion. It may not be backed up by others and you have no way of qualifying the quality of their answers.
- Smart end users can support themselves. Once they invest the time. The first question the end user must ask is, "what is my internal cost per hour?" Followed by, "can a commercial support organization do it more time effectively and more cost effectively?" Finally by; "is this really the best use of my employee's time?" Usually the answer on the second question is yes on both counts, or the commercial support organization does not have business model. It has to be better than you and faster than you. The customer has to be satisfied each and every time the support service is delivered, or the next time the customer will go elsewhere. The answer to the first question is Adam Smith telling us his first rule of open markets is still at work. Price is still closely aligned with cost.
- Open source can allow local suppliers to set up shop easily and offer lower cost local support options in the local language and currency. This mitigates some travel expenses, avoids the delays in expertise arriving on site, and encourages small companies, in limited geographies to try a locally supported but globally available open source product. It does not seem as risky. Larger global companies are prepared to pay for consistent quality, support in depth, and as a result they go with the perceived leaders in the support community. The leaders have to establish a global reputation. One negative comment in a newsgroup can have adverse consequences.

Conclusion

Open source is correcting the imbalance between costs and pricing in the market for widely used, standards based software, in well understood problem domains, and where there is little opportunity for differentiation. The open source genie is now out of the lamp. The opportunity for higher profits has shifted to more of a service based model. This was inevitable. Open source products will be dominant over the long haul in the volume market. Rapidly growing economies in Asia and South America will embrace and incorporate the lower cost model into their infrastructure. The volume of their adoptions alone will overwhelm other numbers. Proprietary vendors will find themselves in a niche market or will be scrambling to reinvent their business model. Just open-sourcing their products will not work. They need to understand where the opportunity in their installed base exists to add value, and where to concede value to the open source products.

Copyright information

© 2005 by Malcolm D. Spence

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

About the author

Malcolm Spence has a Diploma in Aeronautical Engineering from Kingston upon Hull College of Technology, did post graduate study at the College of Aeronautics, Cranfield (both in England), and then obtained his MSC in Civil Engineering (structures) at UMR. He also attended jet engine school at Rolls Royce (UK) and the Tuck Executive Management course at Dartmouth. Malcolm has directed the open source line of business at Object Computing for the last five years. Prior to that, for a year, he was a freelance marketing consultant with various start-up companies. From 1981 to 1998 he was with St Louis sales office, and then the industry marketing group, at Digital Equipment Corporation in Boston. His later assignments involved providing marketing support for DEC Central Engineering OO initiatives in the Asia Pacific region. Before joining DEC, for thirteen years, he was an aeronautical engineer with the McDonnell Aircraft Division of McDonnell Douglas in St. Louis. He resides in the St. Louis area with his wife and three children.

Free software gets small

The mouse that might roar

Christian Einfeldt

Clayton Christensen talks

When Clayton Christensen talks, CEOs listen. Christensen is a Harvard School of Business Associate Professor who has CEOs re-thinking their growth strategies with his concept of disruptive technologies.

Now Christensen is talking about free software gaining a greater market share by getting small. More specifically, by getting small and onto handheld devices. He suggested that Microsoft should seriously consider getting into the Linux space by buying or building a separate company to pursue disruption on BlackBerry-like devices.

Simply put, a disruptive technology, in Christensen's terms, is a business model (like Sony's use of transistors in handheld radios in the early 1960s) that taps a new market in ways that are not interesting to the market leader in any given industry.

If Harvard Professor Christensen is right, tiny Softmaker just might turn out to be the mouse that roared

For example, Honda's little SuperCub Motorcycles dethroned market leader Harley Davidson, who had sold big bikes to North Americans who used them to replace cars as general-purpose transportation vehicles. Harley Davidson's customers rejected Honda's leaky little bikes, which broke down on trips out of town. Honda was on the verge of scrapping the SuperCub for North America when it found in the early 1960s that people would purchase its SuperCubs from sporting goods stores, where the motorcycles were sold as off-road vehicles. Within 20 years, Honda owned the North American motorcycle market, as Honda slowly eroded Harley Davidson's market share with low-cost alternatives. Harley Davidson has never recovered its former glory.

Christensen has recently forecast a similar potential market share drop for desktop software market king Microsoft. And like the case of the Sony transistor radios or the Honda SuperCub, the market entrant, GNU/Linux, will find shelter off the beaten path, and won't make an initial frontal assault on the king. Christensen said that:

> "Where Linux takes root is in new applications, like Web servers and handheld devices. As those get better, applications will get sucked off the desktop onto the Internet, and that's what will undo Microsoft. As computing becomes Internet-centric, rather than LAN (local-area network)-centric, their stuff runs on Linux, because it's all new. As the BlackBerry becomes more capable, applications will get sucked onto it. Those are the kind of places where growth is. If Microsoft catches it, they'll be all right." (Source: Advice to Microsoft: Learn to Love Linux," by Martin LaMonica, Cnet news.com, 2004/10/15, http://news.com.com/Advice+to+Microsoft+

Learn+to+love+Linux/2100-7344_3-5411843.html)

There are two disruptive technologies incubating out there that form the basis of an interesting partnership: OpenOffice.org (OOo), the free software alternative to Microsoft Office (MSO), and Softmaker, a small German start-up which is offering proprietary office solutions for the handheld and the desktop.

Will the mouse roar?

OpenOffice.org by itself represents both a low-end disruption and an emerging market disruption to Microsoft's market-leading MSO product. OpenOffice.org can be used by overshot customers in developing countries, who have the ability, but not the desire, to spend money for MSO upgrades. It can also be used by non-consumers in developing countries, where the cost of MSO means that in many cases, the users' only choices are a pirated version of MSO or OOo. StarOffice, Sun Microsystems' commercial brother of OpenOffice.org, also offers a low-end disruption for companies that want the reliability and name brand of a deep pocket, but don't want or need the greater integration with Windows offered by MSO.

Softmaker, the German start-up, offers format compatibility currently with MSO, and its CEO, Martin Kotulla, has publicly stated that the company plans to release filters for OOo, and has made significant progress along those lines

Softmaker, the German start-up, offers format compatibility currently with MSO, and its CEO, Martin Kotulla, has publicly stated that the company plans to release filters for OOo, and has made significant progress along those lines. Mr Kotulla displayed his Textmaker and Planmaker products, which are text processing and spreadsheet apps, respectively, at the 2004 Desktop Linux Summit in San Diego. Among other things, he touted his products' small executable footprint of four MB.

Microsoft does have a version of Word and Excel for handhelds, but not for handhelds running Linux. OpenOffice.org has announced that a "Lite" version of its widespread office suite will be made part of its global marketing strategic plan; but the OOo Lite version is still in the planning stages, and there is no incubator project for OOo Lite at the time of this writing. The birth of OOo Lite therefore doesn't look particularly imminent.

So the option of OOo working with Softmaker to solve this problem seems like a classic example of how a free software project can work well with a proprietary program to solve a problem

If Softmaker does, in fact, release a OOo/SO filters, Softmaker's products seem to offer a natural fit for Sun's clear interest in competing for the desktop Linux space, and might offer a viable alternative to the proposed OOo/SO Lite. Sun has made its intent clear: to move into the desktop Linux space, for example, with Scott McNealy's announcement at Comdex 2003 that it had secured a deal (http://sfgate.com/cgi-bin/article.cgi?f=/c/a/2003/11/18/BUGRM34CUC1.DTL) with the government of China and the China Standard Software Co. Ltd., for distribution of the Java Desktop System code in the world's most heavily populated nation. In June 2004, Sun has also announced that it had won a deal (http://www.zdnet.com.au/insight/toolkit/businessapps/opensource/0,39024866,39149910,00.htm) for the distribution of StarOffice seats in 72 public and parochial schools in the Canadian Province of Ontario, covering a total of 2.5 million students there. Students in Germany have, for several years now, received instruction in the use of both Microsoft Office and StarOffice.

Sun is not alone in eyeing the free software desktop, of course. Novell, which recently acquired both Ximian and SuSE, is also positioning itself for strong desktop competition. Novell has contributed code to the OOo project, and quite far along in switching to using OOo as its default office suite in-house. Hewlett-Packard has recently announced the availability of SuSE GNU/Linux pre-installed on its new nx5000 notebook, which includes OOo. Linspire has partnered with a Canadian start-up called Sub500.com for the delivery of Linspire's distro on competitively priced desk-

tops and notebooks, also with OOo preinstalled.

So the option of OOo working with Softmaker to solve this problem seems like a classic example of how a free software project can work well with a proprietary program to solve a problem. Here, both Softmaker's products and OOo/SO represent low-end disruptive technologies from the Microsoft point of view.

From Softmaker's point of view, OOo represents a *sustaining* technology, once it gets filters in place for its PDA-capable products. Softmaker's best customers (including me) will be willing to pay a premium for the functionality of being able to use files created with the world's dominant free software suite (OOo/SO) on a handheld, because OOo/SO are currently "not good enough" for that application, as they are too big.

There are tens of million of OOo/SO users already, and that number continues to grow rapidly. Once the OOo filters are in place, Softmaker stands to find a ready market of non-consumers in that emerging market who have pent-up demand for using OOo on their handhelds. I am a classic case in point. I refuse to buy a handheld until I can get my free software files working on it. Period! Lots of other applications currently work on handhelds, but not OOo. I and a million or so of my best friends are just sitting back, waiting for the opportunity to view, modify and save our files on a convenient handheld device, because our batteries on our notebooks are limited, and a notebook is too heavy to lug around. Yet I need my data in court, on planes, taxis, buses, anywhere I have significant down time.

Currently, I just grumble and do without access to my data, because I don't want to use MS Word/Excel on a handheld, for a wide variety of reasons, which basically boil down to the fact that I am too cheap to purchase a copy of the Microsoft products for that purpose, and of course copying without paying is not an option. So I'll wait.

In fact, pursuing non-consumption on handhelds would be directly in line with OOo's recently announced marketing plan:

> "Try to compete against non-consumption: customers who are currently unable to use currently available products at all, either because they can't afford them or are too inexperienced to use them. These markets have the most potential because these customers will compare your product to having nothing at all, and so will be thrilled to buy it even if it's inferior to currently available products." (Source: Clay Christensen and Michael Raynor, *The Innovator's Solution*, p. 11, Harvard Business School Publishing Corp., 2003.)

Also, Microsoft doesn't have a lock on the handheld market as they do with the desktop. This is a real opportunity for Softmaker/OOo/SO to gain market share, because each of those products has cross-platform capability. A partnership like Softmaker/OOo/SO has a real opportunity to get the lead on handhelds.

If Grameen can do it, then why not a partnered team of Softmaker and Sun? Or Softmaker and Novell? Or Softmaker and IBM or HP? Or Sun and Sun?

There's a precedent in Bangladesh

There is a precedent for this kind of emerging market disruptive growth: the case of Grameen Telecom in Bangladesh. This is an extensive quote from the Christensen team's most recent book, *Seeing What's Next:*

> "While most of us in the developed world take our phone service for granted, almost half of the world lacks reliable telecommunications service. In some areas, the cost to build or extend a wireline infrastructure is prohibitive. It would require charging people close to their annual incomes to make telephone calls. Enter disruption."
>
> "Where most firms would see a hopeless situation, Grameen Telecom saw an opportunity. Its approach? Bring wireless service to the country's poor. It loaned up to $175.00 each to women who became known as the 'wireless women of Grameen.' The loan covered the mobile phone's cost, training, and a small solar recharging unit. The women could then sell phone usage on a per-call basis at an affordable price to other villagers."
>
> "Users love the service. It is convenient and cheap. It allows farmers to get vital information such as crop prices without travelling great

> distances. They can place orders with distributors without relying on the postal system. Estimates of per-call savings are up to $10.00 - almost 10 percent of household monthly income. The program has other benefits as well. People travel less, saving wear and tear on shoddy roads [and the travellers' meagre cars and gasoline supplies too]. Phone operators earn $300.00 a year. They plough this money back into education and health care, increasing the country's human capital. Their status in the village increases considerably."
>
> "So the operators are happy and the users are happy. What about Grameen? Its business model allows it be *very* profitable."
>
> (Clayton Christensen, Scott Anthony, and Erik Roth, "Seeing What's Next", Harvard Business School Publishing Corporation, 2004, pp. 219-220)

Of the world's six billion people, five billion live in countries with average annual incomes under US$10,000.00. That provides a huge opportunity for a someone to address the needs of a group like the "wireless women" to sell access to farmers who not only want to phone in orders, but want to keep track of their orders from year to year, so they don't have to reinvent the wheel. If Grameen can do it, then why not a partnered team of Softmaker and Sun? Or Softmaker and Novell? Or Softmaker and IBM or HP? Or Sun and Sun? etc.

Conclusion

While the critics of Sun's deals argue the significance of the bottom-line results from Sun's successes abroad with StarOffice and JDS, these Sun deals do lay the foundation for linkage between Sun's core infrastructure offerings with a more dynamic presence on the desktop. The immediate presence of tiny Softmaker on the handheld space provides at least one plausible case scenario for Sun, or another major Linux player to provide the key linkage with a handheld office package. Softmaker's product is itself, currently proprietary and closed, but Sun has shown an ability and an affinity for acquiring a proprietary solution and either opening the code, as Sun did with OOo (which it bought from German start up Star Division); or seeding diverse applications with easily obtainable code such as its Java code, which can be found on everything from PDAs up to enterprise applications.

If Harvard Professor Christensen is right, tiny Softmaker just might turn out to be the mouse that roared.

Bibliography

[1] The story on Sun's deal with the Ontario schools (http://www.zdnet.com.au/insight/toolkit/businessapps/opensource/0,39024866,39149910,00.htm).

[2] The story on Sun's JDS deal with China (http://sfgate.com/cgi-bin/article.cgi?f=/c/a/2003/11/18/BUGRM34CUC1.DTL).

Copyright information

© 2005 by Christian Einfeldt

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is available at http://www.gnu.org/copyleft/fdl.html.

About the author

Christian Einfeldt is the producer of the upcoming film "the Digital Tipping Point: the culture of freedom in cyberspace", to be released in September, 2005. For more info, go here (http://www.digitaltippingpoint.com)

Free as in “free speech” or free as in “free labour”?

A philosophical enquiry into free culture

David M. Berry

Free software sustains and enables the internet. Across the world, people continue to freely contribute ideas and expertise to an important and growing movement. The internet itself was largely born out of a culture of contributing code and content in an electronic public “space” of global proportions. This has meant that the constellation of software supporting the internet, and the content that sits upon it, is to a large degree, non-market, peer-produced and free (as in “freedom” *and* as in “beer”). But, why do people code, hack, test, write and create free culture?

None of the current theories give a satisfactory answer to the questions raised, namely: why do coders, users and lots of amateur artists contribute to free culture projects?

Free labour sustains the free software and free culture movements, and yet we’re still pretty much in the dark as to why people do it. Indeed, this question continues to puzzle many people who find it difficult to fit into traditional concepts related to production. Theorists and researchers have used a number of approaches: from individualistic or psychologistic theories, which usually end up identifying personal preferences or “motivations”; to concepts of frictionless information or institutional frameworks; essentialist claims about humans; gift economics; free-market economics; bazaars; theories of public-goods; and even the concept of “fun” itself, in an attempt to explain it. But ultimately, the explanatory factors continue to be shrouded in mystery. None of the current theories give a satisfactory answer to the questions raised, namely: why do coders, users and lots of amateur artists contribute to free culture projects?

This article examines free software and free culture, together with the concepts that Hannah Arendt presents in *The Human Condition*, to outline some of the key aspects of labour within what Yochai Benkler calls “peer-produced commons production”. It questions whether there is more than just simple motivational factors at work underneath the surface of the so-called “Hacker Ethic”. Indeed, making “things” is usually classified outside the realm of the life world, and the question of free-software problematises some of our common-sense impressions of what work *qua* work actually is.

Free as in “software”

The dynamics of software production are complex, and the resources required to sustain it are expensive, in terms of equipment and basic necessities, but also in terms of the wider economics of the computing industry. Software development is hugely profitable and employs a large number of people involved in labour intensive, complex and demanding work, which is exploited using copyright and patent monopolies. This, in turn, generates large profit mar-

gins and, as can be seen from salary levels, rewards many programmers handsomely. However, software programming remains a labouring activity; in other words, the activity undertaken by the programmer is exchanged for their wages. The product of this labour, *code*, is alienated and appropriated by the employing company.

Free software, on the other hand, is produced outside of the office-factory - indeed, it is freely produced and contributed to projects that have many features in common. They are owned, if they can be said to be owned at all, by everyone, and it is freely available to be taken and used – although sometimes restricted by license requirements, such as the GNU General Public License (GPL). This code is produced neither wholly as a job nor as a hobby. But its commons-based approach means that it is never wholly appropriated or alienated from the producer either.

This raises two related issues about contemporary free software development that this article intends to examine: (1) Why free-software is undertaken by programmers; and (2) whether free-software points toward a different relationship with the "work of our hands" and the possibilities for emancipation from necessity.

Hannah Arendt's distinctions between *Labor* and *Work* offers useful concepts that can help in understanding why free software is produced and can contribute to further understanding of the activities associated with it. As Arendt explains in *The Human Condition*, the etymology of the two words, *Labor* and *Work*, casts light on the fact that although today we use them synonymously, this wasn't always so. John Locke, for example, drew a distinction between "working hands" and "laboring bodies" and this has some resemblance to the ancient Greek distinction between "the craftsman" and the work of "slaves". This bifurcation of the process of production has also been mirrored in the distinction between manual and intellectual labour and the corresponding connotations and values we have associated between them.

Arendt explains that in classical times, *Labor* was associated with contempt, an activity that left no trace, no monument, no great work worthy of remembrance. Labouring was for those who, like "slaves and tame animals with their bodies, minister to the necessities of life". In order that we could become *political* - i.e. that which distinguished us from animals – we had to escape from necessity. Today, the production of labour is the production of the masses, and Marx notwithstanding, it has historically been feared by the owning or thinking classes – controlled and channelled through overcoding structures that have been able to appropriate their productivity. Think, for example, of the school, the hospital and the assembly lines for the mass produced cars, producing what Foucault termed the *docile bodies* of the workers. These workers performed a fraction of the entire process of production and were thereby alienated from their labour. By being forced to work by necessity (i.e. to enable them to buy food and shelter) meant that they were not able to become fully human; alienated from the products of their own hands, unable to act or think politically and easy to control.

In contrast, Arendt argues that for those who *Work* – who "work upon" rather than "labor and mix with" – there is the possibility for the beginnings of reflexive behaviour (i.e. becoming self-conscious acting beings). *Work* for Arendt, produces durability, the products of work do not disappear but give our common lives stability and solidity – think, for example, of a table passed down generation to generation, contrasted with labouring in a field which would leave no such trace. For Plato, *Poiesis*, as making or fabrication, showed that the craftsman had within his mind an *idea* that could be shaped to a conscious material design: a creative act of production in the physical world. Where *Labor* only produces for the consumption or necessity of the labourer, *Work* creates something durable that will last (for now leaving aside the problematic ontological question of the status of source-code's obduracy). This is important because it is only by escaping necessity (i.e. the constant requirement to produce things we need) that we can begin to communicate and become human as political animals. For Arendt, *Work* is a prerequisite for the possibility of *Action* – the realm of great deeds and great words. As memorably related by Homer, Achilles was remembered for his *Action* - not for his need to satisfy his hunger and clothe his body.

To work for necessity, as many of us do in contemporary consumer society, would be classified by Arendt as *labor*. In fact, Arendt argues that we have become a *laboring society*; in other words, we have succeeded in levelling all human activities to the common denominator of securing the necessities of life and providing for their abundance. Therefore, whatever we do, we do for the sake of "making a living". Anything that cannot be classified as part of making a living becomes a "hobby", subsumed under playfulness

Fig. 1: 'Arendt' art by Trine Bjørkmann Andreassen

or non-serious activities. Our society is now shaped by the importance of producing objects for consumption, and the drive for profitability encourages the production of objects that simply wear out as quickly as possible. Living for necessity was the life of animals – the Greeks even derive their word for torture from *necessity* (the realm of labour) and not from *violence* (the realm of war) and in ancient societies torture could not be applied to *Free Men*, it could only be applied to labouring men or slaves who were subject to necessity anyway.

> We have become a labouring society; in other words, we have succeeded in levelling all human activities to the common denominator of securing the necessities of life and providing for their abundance

Because life depends upon it, necessity is an extremely powerful drive. By directing our labour towards necessity, capitalism is able to reproduce itself as a social system. But free software seems more akin to the realm of *Work* than the realm of *Labor*, it lies to a certain extent outside of the market. Its commons-based production and the action of its producers seem to be less concerned with necessity than building durable, lasting things (the code). The care that goes into much free software production contributes to its craftsmanship – the production of code is in many ways "public", and the source code can be read and admired by others, in this respect it seems to be similar to a great speech or a work of art.

Free as in "freedom"

The second question this article asks is whether free software points toward a different relationship with the work of our hands and the possibilities for emancipation from necessity. Indeed, free software is not directly linked to necessity, and is in many ways similar to the creation of an artist – who Arendt identified as the only real "worker" left in society. She argued that we will not be free until we realise that we are subject to necessity and liberate ourselves from it. Whilst we are forced to "make a living" we will always be caught in a never-ending spiral of *laboring* and consuming. As technology creates more "spare time" the resulting shaping of our desires by the advertising industry makes us greedier and more craving of our appetites - met only by consumption of the consumer goods before us. Free software is created here, strangely enough in the space of consumption. However, it differs, as it is productive and creative. To create free culture is to contribute toward culture rather than consume (i.e. destroy it).

The possibilities offered by free culture are not (yet) completely linked or mediated through the operation of corporations and necessity. People can still write code, blog and share their thoughts on the web, and this act of sharing is also one of communication. However, it is a fragile space that seems similar to the temporary autonomous zones (TAZ) popularised by Hakim Bey. Will the creators of free culture and free software allow it to be overcoded, controlled and channelled towards consumption? Clearly, the vast resources that corporations bring to bear on projects often crush resistance to a bureaucratic mentality. Inevitably, the creators often have different sentiments to those who seek to make money, and eventually they come into conflict with the instrumental rationality of corporations (who aim for profit maximising). Some worrying examples include: the continuing commercialisation of the internet; IBM's forays into the Linux kernel; and even the colonisation of blogging and photosharing.

If we begin to view free software and open source no longer

purely for necessity or instrumentally (i.e. as not just a technical activity) we can reposition it within the realm of human creativity. If free software lies outside the sphere of *labor*, following Arendt, then perhaps, we can begin to understand it as a possible prerequisite for the beginning of political activity. Free software is interesting in that it seems to contribute towards the conditions of possibility for *Work* and *Action* – for humans to perform great deeds, and through the code to speak and create a trace or memory. For instance, Mathew Fuller's concept of *Critical Software* aims to act politically, to subvert existing codes and to give agency and freedom to the usually passive user. Similarly, free software, by giving away the source-code, simultaneously gives agency to the producer and the user opening the possibilities for *Action* rather than directing and controlling the user – e.g. think of the way a word processor can control the user by "suggesting" spellings and grammar.

The realm of economics and markets is the sphere of necessity – you do not have the freedom to act, creatively or politically as an agent. Conversely, free software and free culture seem to be constituted communicatively (i.e. as a conversation between volunteers), and could, therefore, open the possibility for decentralised, non-market commons-based production, within which may lie the seeds of a new politics – the politics of the commons.

Copyright information

© 2005 by David M. Berry

This article is made available under the "Attribution-Share-alike" Creative Commons License 2.0 available from http://creativecommons.org/licenses/by-sa/2.0/.

About the author

David Berry is a researcher at the University of Sussex, UK. His interests include politics, philosophy, intellectual property, immaterial labour, open-source and copyleft. He is also a member of the research collective The Libre Society (http://www.libresociety.org).

Linux Based Solutions
www.snowdoniait.com
Tel: 01248 355 985 Mob: 07976 632 572
SNOWDONIA
IT SERVICES
By using our services you will gain the many benefits of Linux and free software:
• A secure, stable operating system
• A reduction in licensing costs
• A wide variety of applications for home and office use
• A powerful, customisable system
Services available include:
• Installation and configuration of server and desktop systems with appropriate applications
• Training in the use of your Linux system
• Website design with hosting on Linux servers
• Monitoring systems with customised plugins
• Vulnerability scanning and information security programme development
Contact us for a demonstration of Linux and the applications available, or to enquire about our Linux based solutions for home users and businesses.
On-site daytime and evening visits available. Competitive rates, free estimates.

Richard Stallman's blog

Selected entries from Richard's blog (http://www.fsf.org/blogs/rms/).

Richard Stallman

Response to Fox News Article (January 31, 2005)

Jim Prendergast's recent article mistakenly called me a "leader in the open source community". While I appreciate the praise that might be read into that expression, it is not the case: I do not advocate "open source" and never did. I founded the Free Software Movement in 1984. "Free", here refers to freedom, not price; specifically the freedom to redistribute and change the software you use. With free software, the users control the software; with non-free software, the developer has control of the software and its users.

While Microsoft calls GNU/Linux dirty names like "unamerican" and "communist", other presumably capitalist corporations such as IBM and Apple both use and contribute to GNU.

To win freedom, we developed the GNU/Linux operating system, now used on tens of millions of computers. While Microsoft calls GNU/Linux dirty names like "unamerican" and "communist", other presumably capitalist corporations such as IBM and Apple both use and contribute to GNU.

The article misstated my views when it said I am "against intellectual property". That term has no meaning except a confused mishmash of copyright law, patent law, and trademark law, and using the term leads people to simplistic, extreme, confused views. To be either for or against "intellectual property" is equally foolish. We can encourage careful thinking by rejecting that simplistic slogan.

In US law, copyright is a deal between the public and authors: the public sold the freedom to republish, which only publishers could do anyway, and gained more progress. Progress is valuable, but freedoms that we want to use are even more valuable. Nowadays, that includes the freedom to share copies on the internet. To make copyright law a good deal for the public, we should scale it back. If this means some companies and a handful of superstars make less than their wildest dreams, Prendergast may be shocked, but Adam Smith would not have been.

Sincerely,
Richard Stallman
Free Software Foundation

Bolivia (January, 19 2005 to January, 27 2005)

Visiting Bolivia provided an opportunity for me to spend a week with Tania. I had better explain that Tania isn't just my friend; she's my sweetheart. Since she lives in Colombia, it isn't easy for us to spend time together. I used frequent flier miles to get her a ticket to Bolivia while I was to be there. In arranging this trip, I agreed to go speak in Santa Cruz provided either I went there for a very short time or Tania could go with me. The day before going to Bolivia was

when I learned that the trip to Santa Cruz had been arranged for both of us together.

I found that people had arranged a speech for me that afternoon at the vice presidency. The vice president was not at this meeting, but some government officials interested in free software were there

Arriving after an early morning flight which required me to wake up around 4am Bolivia time, I found that people had arranged a speech for me that afternoon at the vice presidency. The vice president was not at this meeting, but some government officials interested in free software were there. (...) (http://www.fsf.org/blogs/rms/entry-20050201)

The next morning we went to the radio station at a university, and I did my interview. Then it was Tania's turn–she founded and manages a free software support company. As she was making her first remarks, I went to plug in my computer. There was a short circuit! The socket exploded in sparks for a second, and then the breaker operated and the whole studio went dark. My whole hand was covered by black soot, and part of my index finger was burnt. But my main thought was that I had just cut off Tania's interview. I felt somewhat ashamed of that. At the same time, the unexpectedness of the situation made me want to laugh.

But Tania had her interview after all. By the time I was able to wash my hand and put some burn cream and a bandage on my finger, they had got the studio running again.

While putting on Saint IGNUcius' tunic at the end, a motion of my arm caused the bandage to fly off my index finger. I said, "Look, a miracle!"

My speech that afternoon was poorly attended. A lot of people had arranged to come to the conference the previous week, when my speech was supposed to be, but only a fraction of them could come at the new date. Nonetheless, it went well. While putting on Saint IGNUcius' tunic at the end, a motion of my arm caused the bandage to fly off my index finger. I said, "Look, a miracle!"

Fig. 1: Valle de la Luna

We went back to La Paz the next day without great difficulty.

On my last day in Bolivia, I went with Tania to see the Valle de la Luna. When it rains, the water goes down through sinkholes into a subterranean river. Some of my photos show a bridge; the bridge was built in a place where it is possible to see down into the subterranean river. The guide told us it received that name after Neil Armstrong visited and said it reminded him of the moon.

Copyright information

© 2005 by Richard Stallman

(The following license is effective immediately)

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

About the author

Richard Stallman is the founder of the GNU Project, launched in 1984 to develop the free software operating system GNU. The name "GNU" is a recursive acronym for "GNU's Not Unix"

Yocto-24

OSS Business Solutions

ATTENTION: SENIOR AND IT EXECUTIVES. GAIN FREEDOM AND SAVE THOUSANDS OF DOLLARS!

ELIMINATE PROPRIETARY SOLUTIONS BY IMPLEMENTING AN OPEN SOURCE SOFTWARE STRATEGY!

RECEIVE A **FREE** OPEN SOURCE KNOWLEDGE STARTER PACK CONTAINING EVERYTHING YOU NEED TO KNOW ABOUT OPEN SOURCE SOFTWARE, ITS BENEFITS AND HOW TO MAKE IT WORK IN YOUR BUSINESS.

CONTACT US **BEFORE 1 APRIL 2005** AND ALSO RECEIVE A FREE OPEN SOURCE ASSESSMENT AND A CONSULTATION SESSION **WORTH $500** FROM YOCTO'S BEST OPEN SOURCE SOFTWARE GURU (GAUTENG, RSA ONLY) WITH NO OBLIGATION AT ALL.

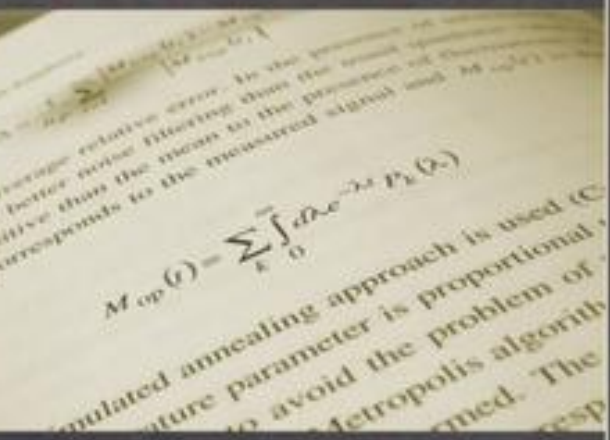
YOCTO OSS HAS THE KNOWLEDGE, EXPERIENCE, AND THE SKILLS YOU REQUIRE TO IMPLEMENT SUCH A STRATEGY. YOCTO OSS PARTNERS WITH MAJOR PLAYERS IN OPEN SOURCE SOFTWARE AND IS ACTIVELY INVOLVED IN THE OPEN SOURCE SOFTWARE RESEARCH COMMUNITY.

CONTACT US VIA OUR WEBSITE HTTP://WWW.YOCTO.CO.ZA OR EMAIL MARKETING@YOCTO.CO.ZA WITH YOUR CONTACT DETAILS AND THE REFERENCE NUMBER FSM001 TO RECEIVE YOUR FREE OPEN SOURCE KNOWLEDGE STARTER PACK.

Make use of open source software to fully support your business strategy.

Do it without any risks to your business!

Communicating mathematics...



...is getting easier.

Math Focus

News
Discussion
Collaboration
Software
Books

www.mathforge.net

Powerful tools for online mathematics

mathforge

Freelock Computing

The Open Source for Business Solutions

Open Source is about solving problems. We solve business problems for small and growing businesses in the Puget Sound area. We provide technology strategy consulting, implementation, and hosting. We work with proven open source software. Go to freelock.com for details.

"In today's world of crappy software vendors who provide crappy products and next to zero service at premium prices, it's refreshing to work with someone who is honest, thorough, reasonable and willing to do what it takes to meet the customer's needs."

– Eric Leung, IS Manager, Outdoor Research, Inc.

We Wrote The Book!

"'Open Source Solutions for Small Business Problems' is a very highly recommended book for anyone who is looking at the open source market ... This is easily one of the best Linux books of the year; providing a management level view of the Linux world without the technical focus of other books."

– Harold McFarland, Readers Preference Reviews

Find an open source service provider or sign up as one!
http://opensourceproviders.com

Contact us at:
info@freelock.com
1-206-579-4836

Freelock LLC
www.freelock.com

qreate

Graphic, Web, Life design

PRINT

Free Software
MAGAZINE
CREATING FREE SOFTWARE MAGAZINE:
A COMPLETE AUTOMATED SYSTEM TO CREATE MAGAZINE BASED ON FREE SOFTWARE
INTERVIEW WITH MYTHIC BEASTS' STAFF
THE SECOND MAC REVOLUTION
THE CONTENT TAIL WAGS THE IT DOG

FREE SOFTWARE MAGAZINE'S
ISSUE 0 COVER

FREE SOFTWARE MAGAZINE'S
MEDIA KIT

FREE SOFTWARE MAGAZINE'S
ADVERTISEMENT

WEB

COLLE MORO BED&BREAKFAST
www.collemoro.com

VILLA DELLE RONDINI
www.villadellerondini.it

www.qreate.net ✳ *info@qreate.net*